Theses and Dissertations 1. Thesis and Dissertation Collection, all items

2012-06

Global Versus Reactive Navigation for Joint UAV-UGV Missions in a Cluttered Environment

Martin, Michael

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/7380

NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

GLOBAL VERSUS REACTIVE NAVIGATION FOR JOINT UAV-UGV MISSIONS IN A CLUTTERED ENVIRONMENT

by

Michael W. Martin

June 2012

Thesis Advisor:	Oleg Yakimenko
Second Reader:	Roberto Cristi

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) WashingtonDC20503.		

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 2012	3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE Global Versus Reactive Navigation for Joint UAV-UGV Missions in a Cluttered Environment		5. FUNDING NUMBERS
6. AUTHOR(S) Michael Martin		

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
  Naval Postgraduate School
  Monterey, CA 93943-5000
		8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number: N/A.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

This thesis presents the coordination of an unmanned, multi-vehicle team that navigates through a congested environment. A novel approach is outlined that enables the control of multiple vehicles based on both computer vision and optimal trajectory algorithms. Various sensors are used to achieve localization in the indoor environment in lieu of global positioning data. Specifically, a Quanser Qball quadrotor is equipped with a downward-looking camera and sonar altimeter, while a Quanser Qbot ground vehicle is outfitted with sonar and infrared range finders. This equipment is complemented by an Optitrack motion-capture system.

Using conventional image-processing techniques, the bird's-eye images supplied by the quadrotor provide information regarding the dynamic environment that surrounds the ground vehicle. The ground vehicle can then produce a global, optimal trajectory, assuring collision-free operations. The optimization problem is addressed by applying the Inverse Dynamics in the Virtual Domain (IDVD) method that uses both the inverse kinematics of the ground vehicle and obstacle information. Furthermore, the IDVD method enables the separation of spatial and temporal planning. As verification of the results of this research, the developed approach for path planning is executed in a fully controlled lab environment and then compared with a sonar-based, reactive obstacle avoidance technique.

14. SUBJECT TERMS: Quadrotor, Ground Vehicle, Image Processing, Trajectory Generation, Navigation.		15. NUMBER OF PAGES 101
		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
  Unclassified
	18. SECURITY CLASSIFICATION OF THIS PAGE
	  Unclassified
		19. SECURITY CLASSIFICATION OF ABSTRACT
		  Unclassified
			20. LIMITATION OF ABSTRACT
			  UU

THIS PAGE INTENTIONALLY LEFT BLANK

GLOBAL VERSUS REACTIVE NAVIGATION FOR JOINT UAV-UGV MISSIONS IN A CLUTTERED ENVIRONMENT

Michael W. Martin
Ensign, United States Navy
B.S., United States Naval Academy, 2011

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

**NAVAL POSTGRADUATE SCHOOL
June 2012**

Author: Michael W. Martin

Approved by: Oleg Yakimenko
                 Thesis Advisor

                 Roberto Cristi
                 Second Reader

                 Knox Millsaps
                 Chair, Department of Mechanical and
                 Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis presents the coordination of an unmanned, multi-vehicle team that navigates through a congested environment. A novel approach is outlined that enables the control of multiple vehicles based on both computer vision and optimal trajectory algorithms. Various sensors are used to achieve localization in the indoor environment in lieu of global positioning data. Specifically, a Quanser Qball quadrotor is equipped with a downward-looking camera and sonar altimeter, while a Quanser Qbot ground vehicle is outfitted with sonar and infrared range finders. This equipment is complemented by an Optitrack motion-capture system.

Using conventional image-processing techniques, the bird's-eye images supplied by the quadrotor provide information regarding the dynamic environment that surrounds the ground vehicle. The ground vehicle can then produce a global, optimal trajectory, assuring collision-free operations. The optimization problem is addressed by applying the Inverse Dynamics in the Virtual Domain (IDVD) method that uses both the inverse kinematics of the ground vehicle and obstacle information. Furthermore, the IDVD method enables the separation of spatial and temporal planning. As verification of the results of this research, the developed approach for path planning is executed in a fully controlled lab environment and then compared with a sonar-based, reactive obstacle avoidance technique.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ATV   All-Terrain Vehicle

DAC   Data Acquisition Card

DCM   Direction Cosine Matrix

DOF   Degrees of Freedom

ESC   Electronic Speed Controller

FLIR   Forward-Looking Infrared

GPS   Global Positioning System

GRASP   General Robotics, Automation, Sensing and Perception

HIL   Hardware in the Loop

I/O   Input/Output

IMU   Inertial Measurement Unit

ISR   Intelligence, Surveillance, and Reconnaissance

LED   Light Emitting Diode

LIDAR   Light Detection and Ranging

LQR   Liner Quadratic Regulator

LSB   Least Significant Bit

LTP   Local Tangent Plane

PID   Proportional Integral Derivative

PWM   Pulse Width Modulation

QuarC   Quanser Real-Time Control

RAM   Random Access Memory

TCPIP   Transmission Control Protocol/Internet Protocol

TTL   Time to Live

UAV   Unmanned Air Vehicle

UGV   Unmanned Ground Vehicle

USB   Universal Serial Bus

USV   Unmanned Surface Vehicle

UUV   Unmanned Underwater Vehicle

UV   Unmanned Vehicle

VTOL   Vertical Takeoff and Landing

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION AND BACKGROUND

A. GENERAL KNOWLEDGE

Advances in science and information technology have brought about the advent of the unmanned vehicle, a powered platform that carries no operator and is either remotely or autonomously controlled. While unmanned vehicles have no single specific use or mission type, they are very versatile and can be used to complete a wide variety of tasks. The vehicles can be designed to be either recoverable or single-use, disposable devices.

Moreover, unmanned vehicles can be organized into four major categories according to the medium through which they travel. These categories are: unmanned air vehicles (UAV), unmanned ground vehicles (UGV), unmanned surface vehicles (USV), and unmanned underwater vehicles (UUV). Military unmanned vehicles are often classified according to the mission type they are assigned. The number of these missions or uses is growing very quickly as leaders begin to appreciate the advantages of having a robot perform tasks that are considered dull, dirty, or dangerous for humans [1].

B. MISSION VARIETIES

Unmanned vehicles are taking the place of humans in many different types of missions. The most recognizable of these tasks are military related. This notoriety is due to the exposure unmanned vehicles receive in the media. Some commonly known missions are mine and bomb detection/disposal, precision strike warfare, and

information warfare. While military missions comprise a large part of all unmanned vehicle tasking, non-military applications of these vehicles are also becoming more prevalent. Some of these missions include search-and-rescue operations, firefighting, and sea-bottom topography. Some of the most notable of all unmanned vehicle tasks, however, are intelligence gathering, reconnaissance, and terrain mapping. Prior to the use of unmanned vehicles, reconnaissance and terrain mapping required the use of an operator who would have to maneuver the vehicle for hours on end, and many times into hostile environments. The confinement and close quarters of these vehicles often caused operator fatigue. Because unmanned vehicles are not encumbered by the physiological limitations that accompany an on-board human pilot, they can be designed to maximize the amount of time a vehicle stays on-station. Unmanned vehicle pilots can exert control over the vehicle remotely and therefore can be swapped in and out as necessary. This capability enables a difficult mission to be completed in a safer and less-fatiguing manner.

C.   VEHICLE TYPES

1.   Air, Ground, and Sea Vehicles

Of all the categories of unmanned vehicles, UAVs have received the most research and attention. Recent successes in various global combat operations have created a demand for UAV technology and a competitive environment for UAV manufacturers. As a result, there are many different types of these aerial vehicles, boasting an impressive range of capabilities. UAVs can be rotary, fixed wing, or floating vehicles. There are many options in the UGV world as well.

These vehicles can be propelled using tracks, wheels, or legs. Maritime vehicles are beginning to receive a significant amount of attention and can either be surface or submersible vessels. For this research, which involves a cooperative mission between two vehicles, a three-wheeled ground vehicle and a quad-rotor were chosen because they each hold significant advantages over other vehicle types in their respective categories.

2. Platform Selection

a. Assessment of Quadrotor

When comparing the quadrotor to other types of aerial vehicles, several distinct advantages and disadvantages come to mind. First, the quadrotor has the ability to hover in one location for an extended period of time. With regard to this research, this fact is a great advantage because the hover capability will help provide a slow, stable platform for the camera mounted on the vehicle. Additionally, the quadrotor's ability to move directly in all three axes allows for maneuverability in tight, constrained places like what might be found in an urban environment. Moreover, the ability to hover allows for vertical take-off and landing, which frees mission planners from the operational constraints imposed by fixed-wing aircraft such as runways. Another benefit of quadrotors is their ability to move directly to an intended location. This enables quadrotors to follow trajectories that would be difficult or impossible for fixed wing aircraft.

Quadrotors are also smaller and less-complicated than many other types of aerial vehicles. Low-power

processors, compact electric motors, and hi-density battery technology allow manufactures to build very small quadrotors. This enhances maneuverability and allows the vehicle to travel in compact spaces. Also, the counter-rotating blades of the quadrotor obviate the need for a tail rotor, which is necessary to counteract the torque produced by the main rotor in a standard helicopter configuration. Quadrotors maneuver by changing the individual speeds of the rotating blades. This fact eliminates the need for a complicated pitch-changing mechanism also used in helicopters.

One of the greatest drawbacks to the quadrotor is its high energy consumption rate. Since the motors are continually operating, mission flight times are limited. Quadrotors, which are usually built smaller to increase maneuverability, have a small payload that restricts the types of sensors the vehicle can carry. These two limitations can play a role in constraining the types of missions available to the quadrotor.

b. Assessment of Ground Vehicle

The chosen ground vehicle for this thesis is the iRobot Create, a vehicle similar in size and shape to the more familiar iRobot Roomba. The vehicle, whose description will be covered in greater detail later, is circular with a wheel radius of 34 centimeters and height of 7 centimeters. One of the greatest advantages of this type of robot over other ground vehicles is its ability to turn 360 degrees while its center stays stationary. This fact, combined with the low height profile of the vehicle, enables it to travel in compact, tight spaces. The vehicle uses a two-wheel

differential drive system with a third omnidirectional caster for balance. This uncomplicated system saves space, provides balance, and allows the motion of the robot to be easily programmed. Finally, the chosen ground vehicle is lightweight, which translates to longer battery life.

While the iRobot is simple and has many advantages, some of the features listed above are also limitations. For example, its low profile and small wheel radii limit the types of terrain the vehicle can transverse. Moreover, the smaller wheels combined with the differential drive system limit the speed of the vehicle.

D. RELATED WORK

1. General

Autonomous vehicle research, and in particular, trajectory generation and computer vision based control, has become a very active area of research. The dramatic increase in successful autonomous vehicle military operations in the last several years has helped to incentivize both commercial and private research in the area. Moreover, coverage on media Internet sites such as YouTube and Facebook has contributed both familiarity and excitement to the field for the general public.

2. University of Singapore

The University of Singapore has a very active UAV research lab that has developed the "Lion" family of UAVs [2]. One member of this family, the SheLion (Figure 1), is a small RC helicopter that has been modified with various add-on sensors such as a camera, sonar altimeter, and inertial navigation system. One of the most beneficial

features of the SheLion is that it has two
processors [3]. One processor is dedicated to handling
image processing operations while the other is solely
dedicated to the flight control of the vehicle. Because
image processing is both time and computationally costly,
by separating these tasks, the creators of SheLion are able
to maintain safer and more efficient results.

Figure 1: "SheLion" UAV and Virtual Model. From [3].

In keeping with the latest trend in small-scale UAV
research, the University of Singapore has been devoting a
significant amount of time to vision-based navigation,
surveillance, and tracking. Using the same UAV helicopter
described above, a real-time vision algorithm was developed
that uses feature extraction to identify likely targets and

a simultaneously-running Kalman filter to estimate and predict the position of the target based on a motion model [4].

3. University of Pennsylvania

One of the most visible quadrotor laboratories in the United States is at the University of Pennsylvania, or UPenn. Some of the most exciting features of the research being done at the General Robotics, Automation, Sensing and Perception Lab (GRASP) at Upenn can be seen on the video-sharing site, YouTube. The recorded videos, which can be found with a simple keyword search, boast millions of views and demonstrate quadrotors performing various aggressive maneuvers. The Grasp Lab has several UAVs which they use to demonstrate their control algorithms and real-time trajectories. One such UAV, the Hummingbird (Figure 2), built by Ascending Technologies, is shown performing flips and a wide array of complex trajectory maneuvers through obstacles and in formation with similar vehicles [5]. This micro-UAV is equipped with four passive optical markers and together with a 20-camera Vicon motion capture system is able to determine its location in the indoor lab environment [6]. The aggressive maneuvers of these quadrotors are possible because the system actively switches controllers during different stages of movement.
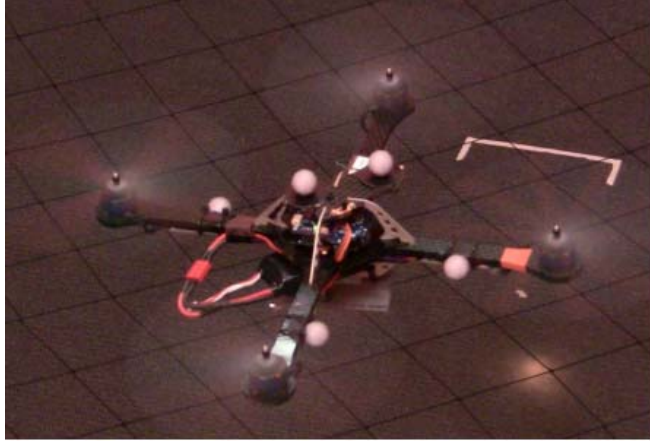
Figure 2: UPenn Micro-Quadrotor. From [6].

One of the key aspects of the research being done at the GRASP lab is the implementation of UAV formation flight. Using a leader-follower approach the micro-UAVs can execute predictive maneuvers and path following. Furthermore, in the event the leader is disabled, the remaining UAVs can estimate the downed vehicle's virtual states [7].

4. Carnagie Mellon University

Researchers at Carnagie Mellon University have also studied the idea of improving the performance of UGVs with global data collected from an aerial vehicle. One project involved using three dimensional overhead LIDAR data to help improve robot localization and global path planning [8]. The autonomous ground vehicle is a modified ATV that is equipped with LIDAR, FLIR, military-grade GPS, and stereo color cameras. One important feature to note is that the terrain data collected by the manned helicopter is downloaded prior to the start of the mission and is not provided real-time to the ground vehicle. The ground

vehicle uses high resolution on-board sensors to help ensure the calculated trajectory remains collision free.

5. DARPA

The Defense Advanced Research Project Agency recently funded a collaborative effort between the Georgia Tech Mobile Robot Laboratory, the GRASP lab, and the University of Southern California's Robotic Embedded Systems Lab to develop a system whereby a single human operator can control a team of aerial and ground robots to navigate through a small village while searching for human targets [9]. The aerial robots, Piper Cub J3 model airplanes, fly above the village taking high resolution aerial pictures while the ground vehicles, consisting of a modified Hummer truck and other assorted smaller vehicles, gather range data. The information is transmitted via an ad-hoc 802.11b wireless Ethernet network. An important feature to note is that this project includes an operator "in the loop" who is an integral component at many stages during the mission.

E. MOTIVATIONS FOR RESEARCH

Unmanned vehicles represent the leading edge of modern day technology. One of the driving motivations behind these systems is the benefit of removing a human operator from tasks that are tedious or dangerous. In the past, many jobs would require vehicle operators to stay on task for multiple hours, which introduces fatigue and a higher likelihood for human error. Unmanned vehicles enable mission organizers to have operators work in shifts to maximize on-scene time as well as performance.

Additionally, since there are no operators on board the vehicle, unmanned systems can be used in environments that are dangerous to human life.

The next step in unmanned vehicle research is to introduce a higher level of autonomy in the system. Autonomous systems reduce the need for human interaction with the vehicle, enabling operators to focus on other tasks. Moreover, autonomous systems can perform at a higher level than a human. On-board processors can collect and integrate data from multiple sensors and respond to this information faster than a human would be able to react. Removing the human element from the vehicle also reduces the need for larger, more armored systems. This reduction enables the lighter, more agile, autonomous vehicles to fly faster and more aggressively than the human body can tolerate.

F. SPECIFIC RESEARCH GOALS

While current unmanned vehicle systems enable the exploration of remote, hazardous areas, they often require a significant amount of human interaction, which is a drain on resources. Furthermore, many current autonomous ground vehicles are designed for reactive navigation using feedback from sensors on the moving vehicle to avoid obstacles. This implementation is time-consuming as the vehicle must gather and process data as it travels. The goal of this thesis will be to design a system architecture that enables the control of a multivehicle team that accomplishes the same mission faster and more accurately. More specifically, a UAV will capture images of the obstacles surrounding the ground vehicle and extract the

location of these obstacles in the global frame. This obstacle information will then be fed to a trajectory generator that will calculate the optimal, collision-free path for the ground vehicle. The process will then be implemented in a controlled lab environment to validate the feasibility of the method.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LAB SETUP

A.   LAYOUT

The laboratory where all trials were conducted is designed so that all experiments can be performed in a controlled, safe environment. Moreover, the lab space can be reconfigured to adapt to the needs of the researcher. The equipment available in the lab consists of multiple Qball-X4 quadrotors, Qbot ground vehicles, an indoor localization system, and two ground stations.

The ground stations consist of PC computers running Windows 7 operating system with 3.20 Ghz processors and 16 GB of RAM. These computers are positioned on the edge of the room and in front of this area is the operating space for the vehicles. The floor is covered by reconfigurable, interlocking rubber mats that are used to reduce glare and reflections that might cause interference with the Optitrack system. This rubber mat also helps protect the vehicles in the event of a system failure.

All lab components can be operated via these ground stations which are equipped with the necessary software. Matlab / Simulink is the primary software in use during all trials. The lab uses QuaRC real-time control software as well as the OptiTrack Tracking Tools package that manages the OptiTrack camera system. Both of these programs are fully integrated with Simulink.

Controlling the vehicles involves running at least two Simulink models on the ground station computer. The host model gathers data from the OptiTrack system as well as the USB joystick that can be used for manual control override.

The model transmits all data to the vehicles using an ad-hoc wireless network. The control models, which are linked to each specific vehicle, compile and download code to the Gumstix processors on board the vehicles.

B.   HARDWARE

1.   Qball-X4

a.   *Introduction*

The Qball-X4 quadrotor vehicle is designed and built by Quanser, a Canadian robotics manufacturing company that specializes in real-time control design. The vehicle has an open-architecture design which allows operators to test a variety of controllers ranging from basic flight dynamics stabilizers to advanced multi-vehicle trajectory planning and navigation algorithms. The vehicle features an on-board data acquisition system, an embedded computer, and a suite of sensors [10]. The quadrotor has a diameter of 0.7 meters and a height of 0.6 meters.

b.   *Protective Cage and Frame*

The Qball-X4 (Figure 3) is a unique quadrotor system because the vehicle platform is enclosed within a spherical, carbon-fiber cage. This feature is ideal for the indoor laboratory environment and helps ensure safe operation during tests that involve vehicles operating in close proximity. The cage is constructed from flexible carbon fiber rods that are inserted into rubber connectors which organize the rods into its spherical shape. The rods are stiff enough to provide support to the quadrotor when it rests on the ground, but are sufficiently flexible enough to absorb the shock of an impact. The rods, which

are easily replaceable, are designed either to break or become loose from the rubber connectors during a crash. The bottom of the cage is truncated slightly to provide the vehicle with a stable landing platform as well as place for the sonar altimeter.

The frame of the vehicle is simply two aluminum cross-beams. The frame provides support to the vehicle components as well as to the cage, which is connected to the frame by rubber mounts to minimize any damage that would be caused by the shock of a collision.

Figure 3: Qball-X4 Vehicle. From [10].

c.   Data Acquisition Card/ Gumstix Processor

The Qball-X4 is outfitted with a HiQ data acquisition card with an embedded Gumstix computer [10]. This assembly, which runs on a Linux-based operating

system, was designed by Quanser to enable vehicle control as well as sensor reading. The HiQ communicates wirelessly with the ground station to relay sensor outputs and receive code designed to be run on the Gumstix computer. The use of Simulink and the Quarc block set allows researchers to build models without intensive programming. Quarc is able to target the Gumstix embedded computer, automatically generate code, and execute the controller. The Input/Output of the HiQ Data Acquisition card (Figure 4) consists of [10]:

- 10 pulse width modulated (PWM) outputs for motor control

- 3-axis gyroscope, with range configurable for ± 75°/s, ±150°/s, ±300°/s, resolution

- 3-axis accelerometer, resolution of 3.33 mg/LSB

- 6 analog inputs, 12-bit, +3.3V

- 3-axis magnetometer, 0.5 mGa/LSB

- 8 channel RF receiver inputs

- 4 Maxbotix sonar inputs

- 2 pressure sensors (absolute and relative pressures)

- 11 reconfigurable digital I/O

- 2 TTL serial ports

- Serial GPS input

Figure 4: Quanser HiQ Data Acquistion Board. From [11].

d. Motors, Propellers, Speed Controllers, and Power

The Qball X-4 is outfitted with four E-flight Park 400 750Kv motors. Attached to each motor are a 10x4.7 propeller (Figure 5) and an electronic speed controller. The speed controllers receive commands from the DAC board in the form of PWM outputs. The Qball-X4 is powered by two 3-cell 2500 mAh Lithium Polymer batteries. These batteries provide a flight time of roughly fifteen minutes and must be maintained at charge above 10.6 volts to avoid permanent damage.

17
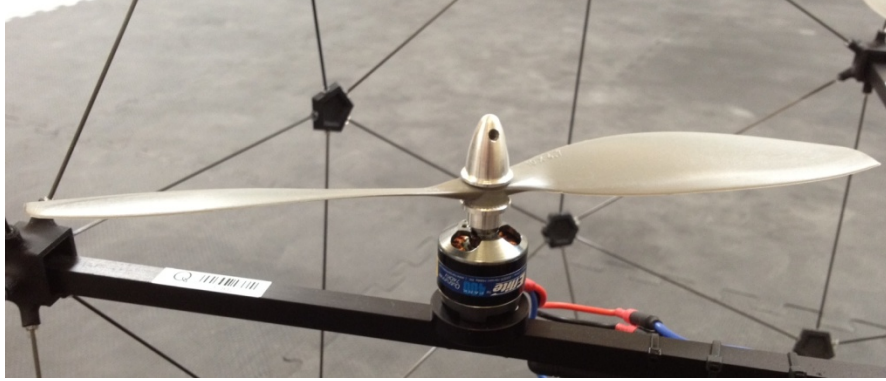
Figure 5: Motor, Propellor, and Speed Controller.

e.   *Sensors and Communication*

The Qball is equipped with several sensors, however, not all of these were used in the control of the vehicle. For example, the magnetometer, which has a listed accuracy of 0.5 mGa/LSB, proved to be unreliable in the indoor laboratory environment. The inconsistency in this sensor's measurements is most likely due to the large amount of unshielded wiring and the construction of the building. As a result, the gyroscope and accelerometer are the sensors chosen to control the roll, pitch, and yaw models of the vehicle. With regard to height control, the sonar altimeter is chosen because it provides very consistent measurements. The sonar used in this experiment is the Maxbotix XL-Maxsonar EZ3. This sonar takes readings at a 10 Hz rate and draws very little current. It has a range of 20–765 centimeters and a resolution of 1 cm [12]. The sonar is fixed to the bottom of the Qball cage so the vehicle pitch and roll must be accounted for in the height control model of the vehicle. Also, a correction must be

made to account for the height difference between where the sonar is located and the center of the body-fixed coordinate frame.

The Qball-X4 is able to achieve localization by means of the external OptiTrack motion capture system. The system uses light emitting diodes and infrared cameras to track the position of passive optical markers (Figure 6) placed on the vehicle. The localization system and method will be discussed in greater detail in a later section.
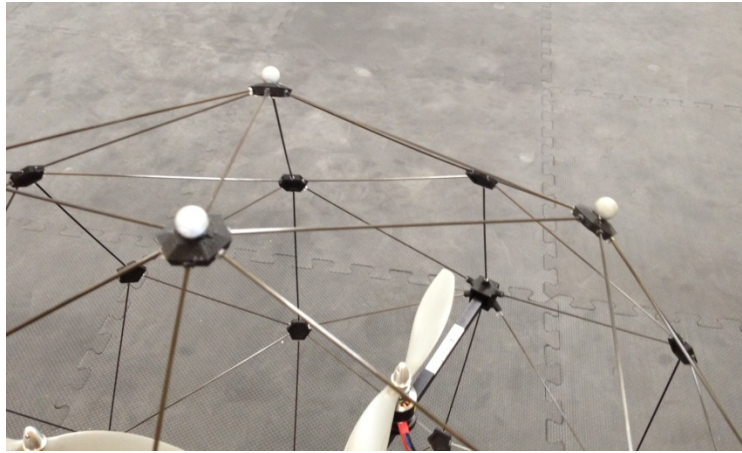
Figure 6: Qball-X4 with Passive Optical Markers Attached.

2.   Qbot

a.   Introduction

The Quanser Qbot (Figure 7) is an autonomous ground vehicle that consists of an iRobot Create and an array of upgraded sensors. The Qbot has been upgraded to include [13]:

- 8 PWM outputs for servo motors

- 7 reconfigurable digital I/O ports, plus 1 digital output LED

- 7 analog inputs, 12-bit, +5V inputs, resolution 6.2 mV

- 5 infra-red (IR) sensors up to 150cm

- 3 sonar sensors 15cm to 6.45m, 1-inch resolution

- 3-axis magnetometer, resolution of 0.77 mGa

- USB camera up to 9fps color images

- Wireless communications

Figure 7: Quanser Qbot. From [14].

b. Frame Design and Drive System

The Qbot has a circular shape with a diameter of 0.34 meters and a height (including camera) of 0.20 meters. With attached sensors, the unit weighs 2.95 kilograms and has a maximum speed of ±0.5 meters/second. The vehicle has

three wheels, two differential drive wheels and one omni-directional caster wheel. The vehicle changes the direction of its movement by altering the speed of the individual wheels.

c. DAC and Gumstix Computer

The DAC aboard the Qbot is located underneath the black cover of the Qbot. Its primary use is to receive any analog or digital inputs that come from the attached or additional optional sensors. The PWM outputs can be used to drive servo actuators like those that might be found in the joints of a robotic arm. The Gumstix embedded processor on the Qbot is similar to that found on the Qball X-4 quadrotor. It is used to compile and execute the code that originates from the Simulink model built on the ground station. The code from the ground station and the sensor feedback from the Qbot are transmitted back and forth using the Wifi board that is connected to the Gumstix.

d. Sensors and Communication

The Qbot is equipped with a variety of sensors. There are three MaxSonar-EZ0 sonar sensors (Figure 8) that are connected to the analog inputs of the Qbot DAC. These sonars cover a range of 6 inches out to 254 inches with a resolution of 1 inch. The SHARP 2Y0A02 infrared sensor is a low cost range sensor that provides measurements in the range of 20–150 centimeters. The Qbot is also equipped with three "bump" sensors that register an analog 1 when the left, front, or middle of the vehicle bumper is pressed.
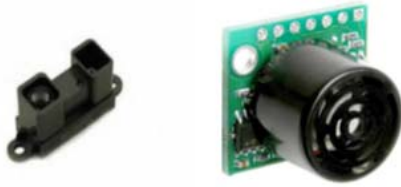
Figure 8: Infrared (Left) and Sonar Sensor (Right). After
        [13].

3.   Optitrack Motion Capture System

In the absence of a GPS signal, indoor laboratories must utilize a different type of localization system to track vehicle position and orientation. This laboratory uses the OptiTrack Infrared Camera system created by Natural Point Incorporated, a company which specializes in optical tracking solutions. There are 10 V100:R2 cameras that have the capability of tracking up to 32 rigid bodies [14]. The cameras were mounted along the ceiling in a manner to eliminate any blind spots in the camera capture volume. The capture volume is the region where the OptiTrack system can successfully track a passive marker. The cubes in Figure 9 represent the approximate capture volume in the lab setup used for this thesis. The pyramids represent the cameras. The capture volume for this lab is approximately 10 feet tall with a width of 12 feet and length of 18 feet.
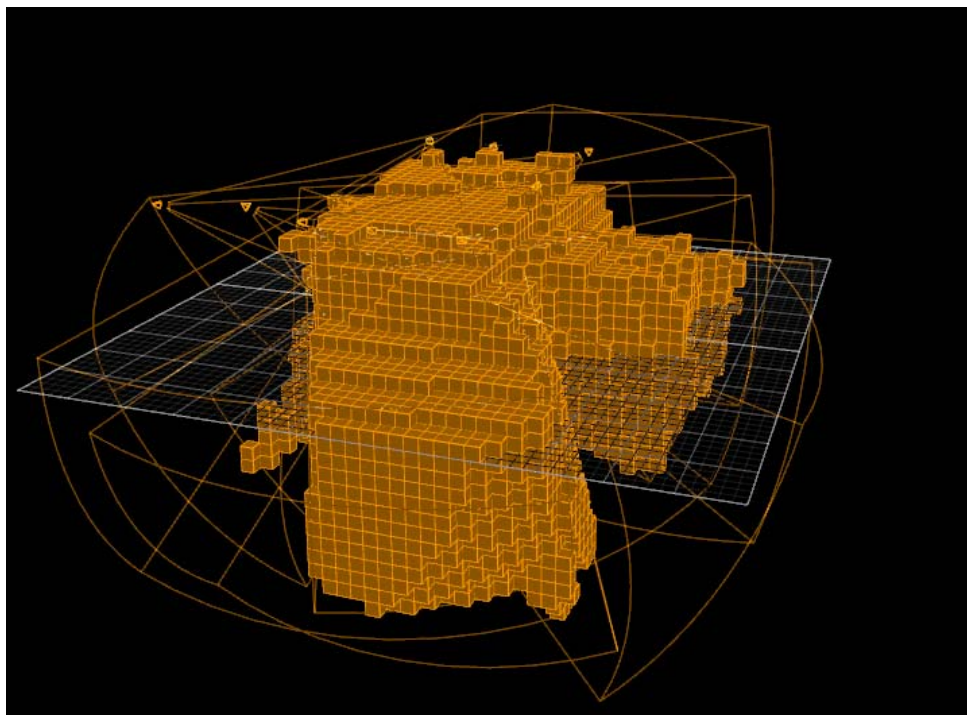
Figure 9: OptiTrack Capture Volume.

The V100:R2 camera (Figure 10) specifications are listed below:

- Resolution-640 x 480

- Frame Rate-100 FPS

- Lens Field of View- 46 degrees

- Interface- USB 2.0

- Latency- 10 milliseconds
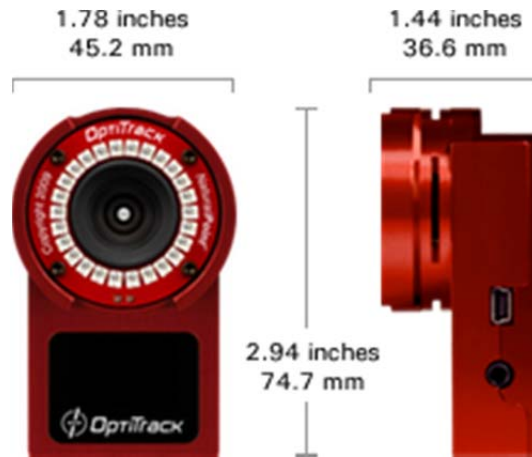
Figure 10:   OptiTrack V100:R2 Infrared Camera. From
                [14].

Each OptiTrack camera is linked to the ground station computer via an OptiTrack USB 2.0 hub. The hubs each link up to five cameras, but each hub must also be connected to other hubs via cable to maintain camera synchronization. The OptiHub configuration is shown in Figure 11.
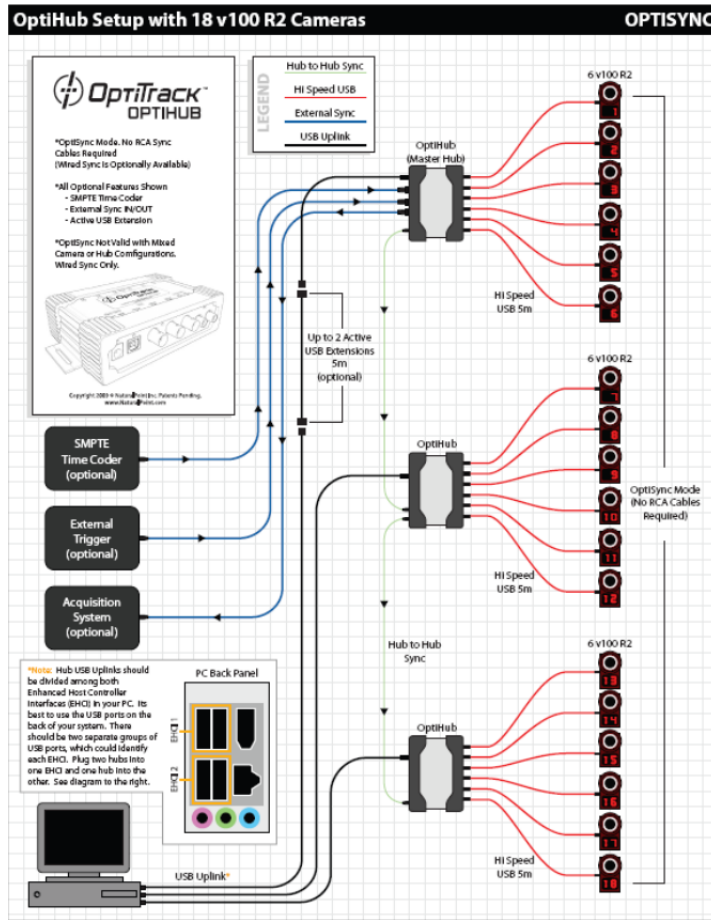
Figure 11:   OptiHub Connection Diagram. From [14].

C.   SOFTWARE

1.   Quanser Quarc Toolbox

Quanser Real-Time Control (QuaRC) software enables rapid control code prototyping and hardware in-the-loop testing. Using QuaRC, a vehicle controller can be built in Simulink and converted to real-time code that can run on many different target processors. This method allows researchers the opportunity to focus on the controller design process without becoming mired in low-level programming. Additionally, an added time-saving benefit of the QuaRC software is that many control parameters in the

model can be adjusted while the code is running. This allows rapid testing and controller tuning without having to recompile the model with each alteration. Another advantage of using the QuaRC software suite is that multiple controllers can be run simultaneously on the same processor. This fact means that one ground station can operate multiple vehicles simultaneously. Since models are run in external mode, real time data can be displayed using scopes and displays, an important feature for vehicle operators.

2. Tracking Tools Software

The OptiTrack Tracking Tools software package is fully integrated with Simulink and the QuaRC toolbox. The QuaRC OptiTrack block set provides the user with the capability of tracking numerous passive optical markers simultaneously in 3-D environment. One of the advantages of the OptiTrack system is that it can be calibrated in roughly five minutes. The calibration process is very simple and only involves the use of two tools, a trident with passive optical markers (Figure 12) on the tips and an L-shaped tool that is used to mark the zero point of the room.

The process involves first performing a visual check of each camera view to ensure there are no false reflections from objects in the camera field of view. If the reflecting object is not easily removable from the workspace, then the software allows one to place a virtual mask over this reflection. After this check is complete, the user begins "wanding" by moving the trident in a figure-eight pattern throughout the entire lab workspace. During this process, the software provides an indication to

the user about the general quality of the calibration. When the desired quality is achieved, the final step is to place the L-shaped ground plane tool on the ground in the center of the capture volume. The software then sets this as the origin from which all measurements will be based. The calibration is them saved to a file which must be referenced in the ground station host model that provides the localization data to each vehicle. Another important feature of the Tracking Tools Software is the ability to create "trackables." These are unique arrangements of passive optical markers that can be fixed to vehicles in order to track not only the vehicle's x-y-z position, but also the angular orientation of the vehicle.
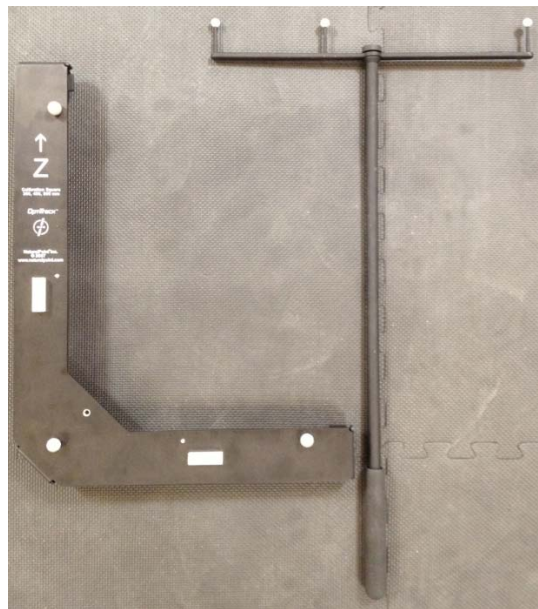
Figure 12:   Ground Plane Tool (left) and Wanding Tool (right).

THIS PAGE INTENTIONALLY LEFT BLANK

III. VEHICLE MODELING AND CONTROL

A.   INTRODUCTION

Prior to performing any obstacle avoidance or trajectory generation, the motion and control of the vehicles must be understood. Feasible collision-avoidance trajectories require knowledge of the physical parameters and correct control inputs for the vehicle. In this chapter, the simplifying assumptions about the operating environment and vehicles will be outlined. This will be followed by a section about coordinate frame designation and sections about the control and modeling of the Qball-X4 and Qbot.

The modeling of the vehicles is outlined according to state space format. This means that the dynamics of each vehicle is contained in a set of differential equations which is represented in matrix format. Certain assumptions are made and linearization about a point occurs.

1.   Assumptions and Simplifications

There are several assumptions that can be made to simplify the complexity of the Qball-X4 and Qbot models:

- The Earth is a flat, non-rotating surface.

- Earth is the only body that exhibits a gravitational force and the acceleration due to this force is a constant 9.81 $\frac{m}{s^2}$.

- The Quadrotor and Qbot have rigid bodies that do not flex.

- Drag forces are negligible due to slow vehicle speeds.

- Quadrotor pitch and roll angles are small.

- The Qbot is symmetric about its centerline axis while the quadrotor is symmetric about both its pitch and roll axes.

2.   Coordinate Systems

With the exception of the image processing coordinate frames that will be discussed later, this thesis deals with two main coordinate frame types. The first type is the body fixed frame. There are two body fixed frames (Figure 13) that attach to the centers of each vehicle and rotate with them. An X-Y-Z Cartesian coordinate system is used for both body fixed frames.
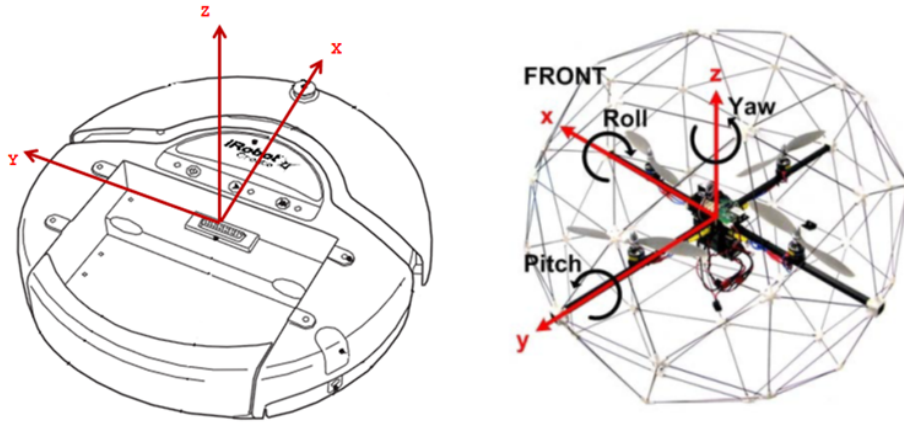
Figure 13:   Body Fixed Coordinate Systems. After [10,13].

As shown in Figure 13, the X axis aligns with direction of forward movement, the Y axis points to the left, and the Z axis points up. All coordinate systems used in this thesis are right-handed coordinate systems. The

other type of coordinate system that is used is a local tangent plane (LTP) coordinate system. This coordinate system is used by the OptiTrack system as a reference coordinate system for the vehicles. The LTP serves to approximate the Earth as a flat, non-moving object. This approximation is possible because neither the Qbot nor the Qball operate at fast speeds or travel long distances. The effects of the Earth's curvature and sidereal motion can be ignored. The X-Z plane is placed on the floor of the lab, while the Y axis point upward.

B.   QBALL-X4 QUADROTOR

1.   Motor Control and Thrust Modeling

A quadrotor is a type of rotorcraft that is propelled by four symmetrically-pitched, but independently controlled rotors. Generally, control of the vehicle is achieved by altering either the pitch of the blades or the individual rotational rates at which they each turn. As discussed previously, the Quanser Qball-X4 does not have any complicated pitch-changing mechanisms so the individual rotor thrusts control the movement of the vehicle. In order to keep the vehicle stable along the yawing axis, the four rotors are organized into two sets of counter-rotating pairs (Figure 14) along the X and Y axes.
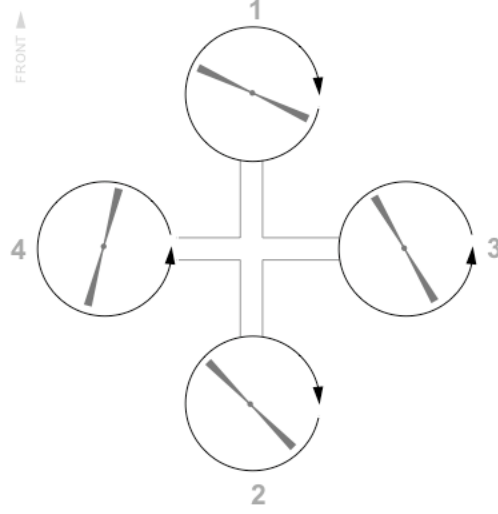
Figure 14:   Quadrotor Motor Dynamics. From [15].

The specific thrust that is generated by each propeller is modeled by the first order system equation

$$F_{thrust,i} = K\frac{\omega}{s+\omega}u_i \qquad (1)$$

In this equation, u_i represents the PWM input of the motor, ω is the bandwidth of the motor, and K is a positive gain. The state variable v_i is $K\dfrac{\omega}{s+\omega}u_i$ and this is used to represent actuator dynamics. The control inputs for each of the specific controllers are defined as:

$$
\begin{aligned}
U_1 &= \frac{F_{T_1} + F_{T_2} + F_{T_3} + F_{T_4}}{m} \\
U_2 &= F_{T_2} - F_{T_4} \\
U_3 &= F_{T_1} - F_{T_3} \\
U_4 &= (F_{T_1} + F_{T_3} - F_{T_2} - F_{T_4})*d
\end{aligned}
\qquad (2)
$$

The value, d, is the force to moment scaling factor that is
dependent on various blade parameters such as Reynolds
number, Mach number, and angle of attack. Some necessary
system parameters are tabulated as:

Parameter	Value
J	0.03 kg·m^2
J_{zz}	0.04 kg·m^2
K	120 N
K_y	4 $N·m$
L	0.2 m
M	1.4 kg
ω	15 rad/s

Table 1.   System Parameters. From [16].

2.   Roll, Pitch, and Yaw Models

Roll and pitch of the quadrotor occur around the X and
Y axes respectively. An ideal feature about the Qball is
that it is symmetric about its X and Y axes. This implies
that the moments of inertia $J_{xx} = J_{yy} = J$, a fact that
simplifies the dynamic equations for the second derivatives
of roll and pitch. Also, because hover flights do not
require large pitch and yaw angles the following
simplification can be made:

$$\phi << 0.1 \rightarrow \sin(\phi) \simeq 0, \cos(\phi) \simeq 1$$
$$\theta << 0.1 \rightarrow \sin(\theta) \simeq 0, \cos(\theta) \simeq 1$$

(3)

Having made these simplifications, the roll, pitch, and yaw
rates can be stated as:

33

$$J_{xx}\ddot{\phi} = U_2 l$$
$$J_{yy}\ddot{\theta} = U_3 l \tag{4}$$
$$J_{zz}\ddot{\psi} = U_4$$

Using $\Delta u = u_1 - u_2$ or $\Delta u = u_3 - u_4$ we can establish state space equations for roll and pitch models (Equations 5 and 6 respectively).

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \dfrac{Kl}{J} \\ 0 & 0 & -\omega \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \Delta u \tag{5}$$

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \dfrac{Kl}{J} \\ 0 & 0 & -\omega \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \Delta u \tag{6}$$

As a result of the symmetry of the Qball-X4, the pitch and yaw controllers are very similar. The controller is a linear quadratic regulator (LQR) that appends a fourth state $\dot{s} = \phi$ and $\dot{s} = \theta$ to allow for integrator feedback.

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{v} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{KL}{J} & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} \Delta u \tag{7}$$

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{v} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{KL}{J} & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} \Delta u \tag{8}$$

The implemented controller is in the state-space form $\dot{X} = Ax + Bu$ where **x** is the state variable, **A** is the state matrix, and **B** is the input matrix. The controls given in the form u=-k**x.** An LQR controller works by minimizing a cost function with weighting factors determined by the control designer. The feedback gain, k, that results from these inputs results in poles at $-19.827, \ -4.083+4.275i, \ -4.083-4.275i, \ -0.316$ and can be determined from the weighting matrices **Q** and **R.**

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 22000 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$
$$R = 30000 \tag{9}$$

With respect to the yaw model, the torque generated by each motor is proportional to the PWM input of that respective motor. As shown previously, $J_{zz}\ddot{\psi} = U_4 = K_y \, \Delta u$ where $\Delta u = u_1 + u_2 + u_3 + u_4$. In state-space form:

$$\begin{bmatrix} \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{K_y}{J_{zz}} \end{bmatrix}\Delta u \tag{10}$$

The LQR controller for the yaw axis was designed in the same manner as the pitch and roll controllers. The weighting matrices are:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}$$
$$R = 1000 \tag{11}$$

3.   Position Model

The quadrotor changes its location in the X-Y plane by altering its pitch and roll angle. This is apparent because when the quadrotor has a non-zero pitch or roll angle, the thrust vectors from the rotors have a horizontal component. Once again, assuming small pitch and roll angles, $\ddot{x}$ and $\ddot{y}$ have the following state space equations.

$$\ddot{x} \approx \frac{1}{m} 4Kv\theta$$
$$\ddot{y} \approx -\frac{1}{m} 4Kv\phi$$

(12)

$$
\begin{bmatrix} \dot{X} \\ \ddot{X} \\ \dot{v} \\ \dot{s} \end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & \dfrac{4K}{m}\theta & 0 \\
0 & 0 & -\omega & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix} X \\ \dot{X} \\ v \\ s \end{bmatrix} +
\begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} u
$$

$$
\begin{bmatrix} \dot{Y} \\ \ddot{Y} \\ \dot{v} \\ \dot{s} \end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & \dfrac{4K}{m}\phi & 0 \\
0 & 0 & -\omega & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix} Y \\ \dot{Y} \\ v \\ s \end{bmatrix} +
\begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} u
$$

(13)

Once again, the controller was designed using LQR methods on Matlab with poles at $-6.712, -1.61 + 0.792i, -1.61 - 0.792i, -0.142$ and with weighting matrices given by:

$$Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = 50 \tag{14}$$

4.   Height Model

Vertical motion is caused by the vertical component of thrust provided by the rotors and can be modeled as:

$$M\dot{z} = 4F\cos(\phi)\cos(\theta) - Mg \tag{15}$$

F, M, and g are the thrust from the propellers, the mass of the quadrotor, and acceleration of gravity respectively. The height controller, unlike the previous yaw, pitch, and roll controllers, is a proportional, integral, derivative (PID) controller. A PID controller calculates an error value as the difference between a measured variable and a desired outcome. By tuning the three gains Kp, Ki, and Kd, the control designer can meet specific outcome requirements. The values for these gains are listed in Table 2.

Gain	Symbol	Value
Proportional	K_p	0.00621
Integral	K_i	0.0015
Derivative	K_d	0.0078

Table 2.   PID Controller Gains

C.   QBOT GROUND VEHICLE

1.   Introduction

As one might expect, the modeling and control of a three-wheel ground vehicle is considerably less involved than that of a quadrotor. Differential drive robots, like the Qbot, have added simplifying features that make them easier to model than other types of wheeled UGVs. The Qbot has two independently-driven, coaxial wheels and a non-powered third wheel that provides stability but freely rotates to prevent wheel side slippage. The powered wheels provide the forward and reverse motion of the vehicle as well as the rotation of the vehicle about its center axis. This rotation is due to the difference in speed between the two wheels, v_{right} and v_{left} (Figure 15). Another simplification is that the Qbot lacks a suspension system and therefore its motion be modeled in just the X-Y plane. With a top speed of $|v_{left}|=|v_{right}|=0.5$ m/s, the Qbot can transverse the very flat lab floor with virtually no vertical movement. Therefore, the Qbot (Figure 15) has three degrees of freedom, the X and Y position of the vehicle and its rotation about the Z-axis.
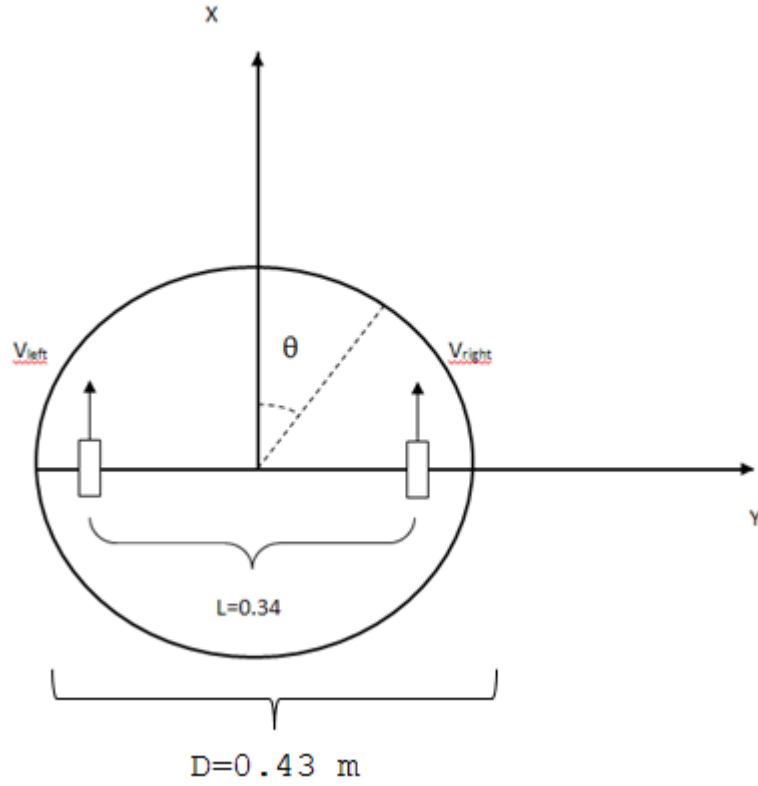
Figure 15:   Qbot Important Parameters

The equations which govern the movement are given by:

$$\dot{P} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dfrac{v_{right} + v_{left}}{2} \cos\theta \\ \dfrac{v_{right} + v_{left}}{2} \sin\theta \\ \dfrac{v_{right} - v_{left}}{L} \end{bmatrix} \tag{16}$$

2.   Inverse Kinematics

As was mentioned in the previous section, the trajectory of the Qbot can be controlled by varying the velocities of the wheels. The controllers implemented in this thesis accept v_{right} and v_{left} as inputs. From the first two

39

equations of Equation 16, the term $v_{right} + v_{left}$ as well as an expression for θ can be determined.

$$v_{right} + v_{left} = 2*\sqrt{\dot{x}^2 + \dot{y}^2} = F_1 \qquad (17)$$

$$\theta = \tan^{-1}\left(\frac{\dot{y}}{\dot{x}}\right) \qquad (18)$$

Differentiating Equation 18 yields:

$$\dot{\theta} = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} \qquad (19)$$

Since the third equation of system (16) states $\dot{\theta} = \dfrac{V_{right} - V_{left}}{L}$, then

$$v_{right} - v_{left} = L\frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} = F_2 \qquad (20)$$

Resolving equations (17) and (20) yields:

$$v_{right} = \frac{F_1 + F_2}{2} = \frac{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}} + 0.5L(\ddot{y}\dot{x} - \ddot{x}\dot{y})}{(\dot{x}^2 + \dot{y}^2)}$$

$$v_{left} = \frac{F_1 - F_2}{2} = \frac{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}} - 0.5L(\ddot{y}\dot{x} - \ddot{x}\dot{y})}{(\dot{x}^2 + \dot{y}^2)} \qquad (21)$$

In the absence of disturbances, knowing the controls v_{right} and v_{left} enables one to know the trajectory x(t), y(t).

IV. REACTIVE MOTION PLANNING

A.   INTRODUCTION

Motion planning refers to the ability of a mobile, autonomous system to plan its motions [17]. One category of autonomous motion planning is called reactive motion planning. This type of planning enables the vehicle to navigate through either a static or dynamically changing environment by detecting information about obstacles using feedback from on-board sensors. The type of motion planning involved in this thesis will be a combination of local reactive planning and global deliberate planning. Specifically, the ground vehicle will be given an end destination, but will need to find its own path through a series of static obstacles using a variant of the potential field technique.

B.   BASICS

The potential field method is a simple, yet effective approach to reactive navigation. The potential field approach was chosen because of its ease of implementation as well as for its directness. This is opposed to other methods like the Voronoi Diagram method, which keeps the robot as far as possible from all obstacles in the workspace and has relatively long path lengths. With regard to potential fields, the target exhibits an attractive potential field, while obstacles have a repulsive potential field. The resultant potential field is computed by summing the contributions from all the potential fields in the environment:

$$U = U_{target+} \sum U_{obstacle} \qquad (22)$$

$$F = -\nabla U \qquad (23)$$

The robot movement direction is then inclined to the direction of the local force [18].

C. VECTOR FIELD

This thesis uses a variation of the potential field method called the vector field method. Like the potential field method, the vector field method uses a virtual force field to direct the robot. This field is based on the robot's sensory perception of the environment and it is the weighted vector summation of the force field that gives the correct vehicle heading. The repulsive force is defined by:

$$\vec{F}_{obs,i} = -w_i^2 \vec{V}_i$$

$$\vec{F}_{obs} = \sum_i \vec{F}_{obs,i} \qquad (24)$$

$$w_i = 1 - \frac{\min(d_{th}, d_i)}{d_{th}}$$

Here $\vec{V}_i$ represents a position vector that gives the range data (d_i) obtained by i[th] infrared sensor. $\vec{F}_{obs,i}$ is the repulsive force that is associated with $\vec{V}_i$. The term d_{th} represents the maximum range of the IR sensors.

Prior to combining these forces to find the appropriate direction vector, each force is converted to a unit force given by:

$$\hat{F}_{t\,ar} = \frac{\vec{V}_{tar}}{\left|\vec{V}_{tar}\right|}$$

$$\hat{F}_{obs} = \frac{\vec{F}_{obs}}{\left|\vec{F}_{obs}\right|} \quad\quad (25)$$

$$\vec{F} = w_{obs}\hat{F}_{obs} + w_{tar}\hat{F}_{tar}$$

In these equations, $\hat{F}_{t\,ar}$ is the unit attractive force that is linked to the target position vector, $\vec{V}_{tar}$. $\hat{F}_{obs}$ is the unit repulsive force. The weights, w_{obs} and w_{tar}, are used to regulate the unit forces to produce the correct direction vector $\vec{F}$. The robot follows $\vec{F}$ for collision free navigation. As will be explained in the last chapter, the experimental trials revealed that there are many sources for error involved in using this method ranging from sensor error to problems that arise based simply on the geometry of the obstacle field, like the local minimum problem. This problem occurs when the robot becomes "stuck" after finding the local minimum of the environment rather than the global minimum.
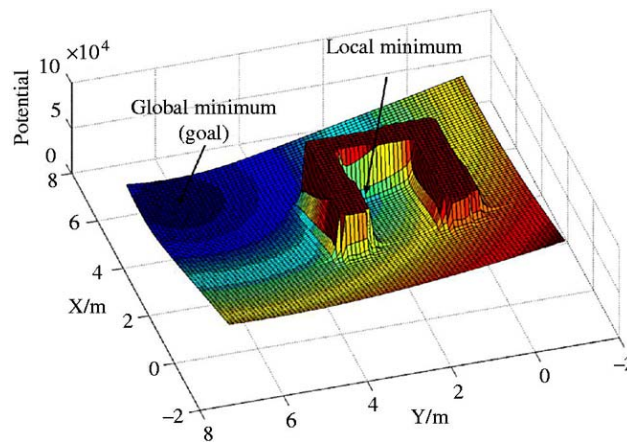
Figure 16:   Local Minimum Problem. From [19].

43

In Figure 16, a robot might follow the force vector into the horseshoe-shaped obstacle and become stuck in the local minimum there. There are numerous strategies and various bug algorithms like wall following that can be used to combat this problem, however, this thesis will not involve complicated geometries with local minima as this will only increase the amount of time required for the robot to traverse the environment using reactive navigation. Additionally, circular obstacles will be used, which also helps to reduce errors in the reactive navigation portion of the thesis.

V. IMAGERY ANALYSIS

A. INTRODUCTION

Computer vision and image processing are very active areas of research with regard to autonomous systems and environment navigation. Computer vision navigation relies on image sensors to provide data to the system about its external surroundings. This process is not always as simple as it sounds as images must first be acquired, processed, and analyzed in order to find out any useful information. This thesis will make use of the image acquisition process known as the pinhole camera model. Moreover, the physical camera will be a downward-looking, wide-angle camera.

A. IMAGE THRESHOLDING AND CENTROID CALCULATION

Image thresholding is a very simple form of image segmentation. From a greyscale image, thresholding can be used to make a binary image. The most important parameter with regard to thresholding is the threshold value. The process by which thresholding is accomplished in this thesis is as follows. First, an initial threshold value of 60% is chosen. The image (Figure 17) is then segmented into object and background pixels. If a pixel is brighter than the threshold limit it is marked as an object pixel and it is given a value of "1". Conversely, if a pixel is darker than the threshold, it is marked as a background pixel and it is given a value of "0". The binary image (Figure 18) is then based off these numbers. In this thesis, the obstacles are white cylinders and the floor is black foam so the image thresholding was accurate and non-problematic. In order to find the centroid of the obstacles, the white

pixels, labeled "1," are grouped together as objects if a certain number of similarly labeled pixels are adjacent to one another. Then the means of the x and y coordinates of this object are calculated and the new point (X_{mean}, Y_{mean}) is called the centroid.
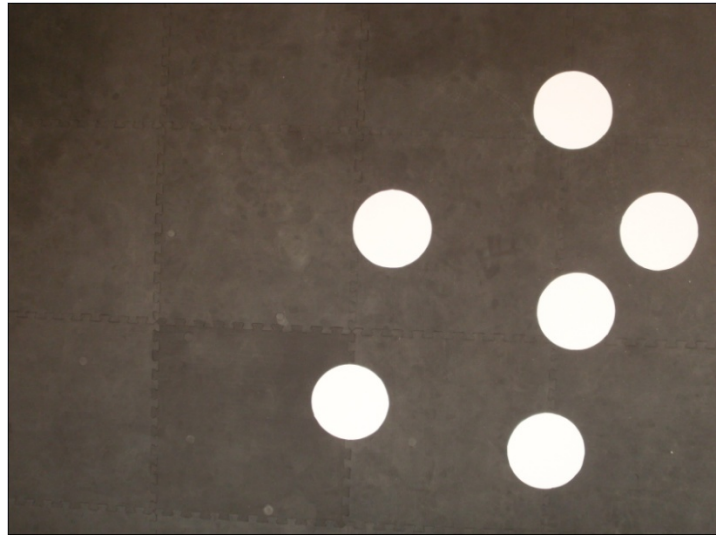

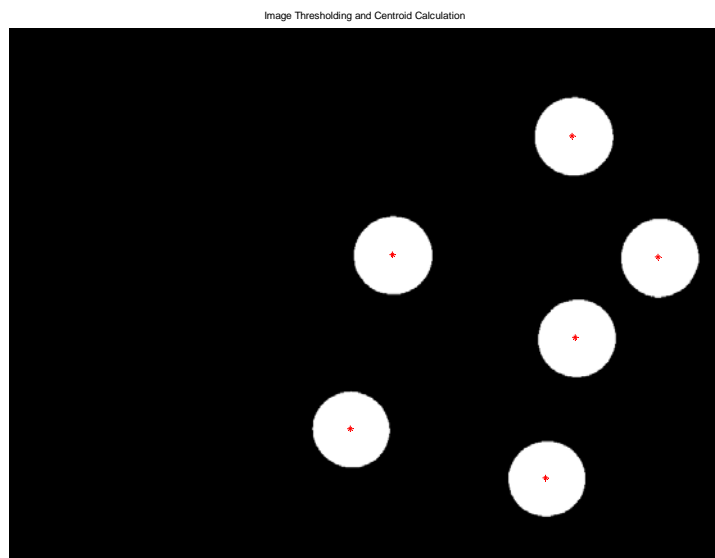Figure 17:   Black and White Image

Figure 18:   Image Thresholding and Centroid Calculation

C.   PINHOLE MODEL AND COORDINATE FRAMES

The pinhole camera is a popular way to model a camera for computer vision processes. As is shown below, the model (Figure 19) involves the use of three coordinate systems: the image, camera, and world coordinate systems. For the pinhole camera depicted in Figure 14, the plane R is the image plane and the location O_C is called the optical center. The distance between R and O_C is called the focal length, f. The location O_I, where the optical axis intersects R, is called the principal point. The coordinate system (O_I, X_I, Y_I) is considered the image coordinate system and the coordinate system (O_C, X_C, Y_C, Z_C) is called the camera coordinate system.

In order to relate a position to a global frame, a world coordinate system (O_W, X_W, Y_W, Z_W) must also be defined. Below is a figure representing these coordinate systems.
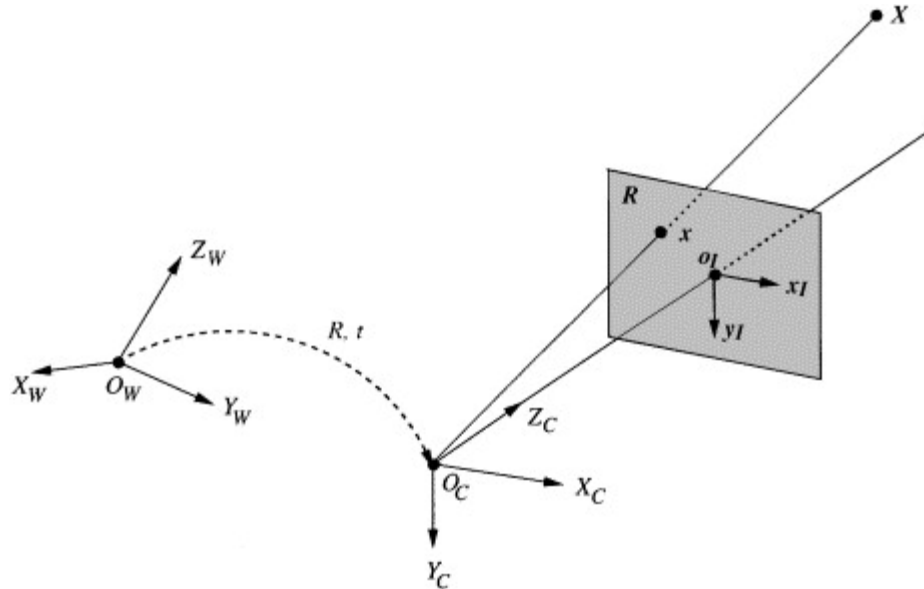

Figure 19:   Pinhole Camera Model. From [20].

Also the points **X** and **x** are defined with coordinates (x_c, y_c, z_c) and (x_i, y_i) respectively. From similar triangles, one can see that in the far field where z>>f a relationship exists between a point in the image frame and a point in the camera frame.

$$x_i = -f \frac{x_c}{z_c}$$

$$y_i = -f \frac{y_c}{z_c}$$

(26)

Therefore, if **x**=P$_f$**X** then in homogenous coordinates:

$$
P_f = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & f \end{bmatrix}
$$

$$
P_f = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & f \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} fx_c \\ fy_c \\ 0 \\ f - z_c \end{bmatrix} = \begin{bmatrix} \dfrac{fx_c}{f - z_c} \\ \dfrac{fy_c}{f - z_c} \\ 0 \\ 1 \end{bmatrix} \simeq \begin{bmatrix} -f \dfrac{x_c}{z_c} \\ -f \dfrac{y_c}{z_c} \\ 0 \\ 1 \end{bmatrix}
$$

(27)

D. TRANSFORMATIONS

Part of the challenge of using a camera as a sensor is the fact that there are multiple coordinate systems that must be resolved in terms of one another. This will be accomplished through a rotation and translation matrix. In this thesis, the shorthand notation $c\theta$ is used to represent $\cos(\theta)$ and similarly $s\theta$ to mean $\sin(\theta)$. To transform vectors

from one coordinate frame to another, the direction cosine matrix (DCM) is used. The rotation matrices about the x, y, and z axis are:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (28)$$

$$R_y = \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (29)$$

$$R_z = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (30)$$

Additionally, a translation term **d** can be added to allow for translational movement from one frame to another.

$$R(\theta,d) = \begin{bmatrix} R_{11} & R_{12} & R_{13} & d_x \\ R_{21} & R_{22} & R_{23} & d_y \\ R_{31} & R_{32} & R_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (31)$$

Given an object "s" with only coordinates x and y (z=0), with the equations above one is able to compute the projection of that object in the world frame on to the image frame.

$$^c p_s = P_f R(\theta, {}^w d_c) {}^w p_s \qquad (32)$$

$$\begin{bmatrix} w{}^{c}x_{s} \\ w{}^{c}y_{s} \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & -1 & f \end{bmatrix} \begin{bmatrix} R^{T}(\theta) & -R^{T}(\theta)*{}^{w}d_{c} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{w}x_{s} \\ {}^{w}y_{s} \\ 0 \\ 1 \end{bmatrix} \tag{33}$$

The term w in this equation is a scaling factor. While Equation 33 presents a useful relationship, in this thesis we are interested in finding the world coordinates of an object based off its image coordinates. Defining M as:

$$M = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & -1 & f \end{bmatrix} R_{3}(\theta, {}^{w}d_{c})$$

$$\begin{bmatrix} {}^{w}x_{s}w' \\ {}^{w}y_{s}w' \\ w' \end{bmatrix} = M^{-1} \begin{bmatrix} {}^{c}x_{s} \\ {}^{c}y_{s} \\ 1 \end{bmatrix} \tag{34}$$

$$w' = M^{-1}(3,1){}^{c}x_{s} + M^{-1}(3,2){}^{c}y_{s} + M^{-1}(3,3)$$

In Equation 34, $R_{3}(\theta, {}^{w}d_{c})$ is $R(\theta, {}^{w}d_{c})$ without the third column. Equation 33 is only invertible when the camera is high in the sky. The notation in the scaling factor equation to find w' refers to $M^{-1}(i,j)$ where i is row and j is the column of the M^{-1} matrix. In this thesis, the objects in the world frame will be the obstacles located on the ground plane where z=0 and the camera will be stationary above this plane at a height of roughly 1.5 meters.

VI. OPTIMAL CONTROL BY DIRECT METHOD

A.   INTRODUCTION

The optimal control problem this thesis addresses is to guide a ground vehicle from an initial state to some final state with constraints imposed on both the states and the controls. Ideally, the routine that is used should be capable of updating itself multiple times over the course of the trajectory to mitigate disturbances and unmodeled motor dynamics. Traditional indirect methods are not able to handle this problem in real-time, leaving the alternative direct method as the ideal choice to formulate our vehicle path.

The main idea behind the direct methods technique is the use of a finite set of variables to arrive at an optimal or quasi-optimal solution. This approach involves using a function to approximate the states and controls and a function to represent the cost of the process. The coefficients of these function approximations are used as variables in the optimization problem, which becomes a nonlinear optimization problem rather than the more challenging boundary-value problem. A great amount of research has been done with regard to this type of optimization and there are a wide variety of optimization software packages available that can be used to solve problems somewhat quickly [21]. However, many of these methods, such as the direct collocation method, rely on thousands of variables and constraints, which add significantly to computational time and cost.

51

The method that is used in this thesis research is the direct method of calculus of variations exploiting the inverse dynamics of a vehicle in the virtual domain (IDVD) [22]. As the name suggests, one of the key features of this method is that it utilizes the inverse dynamics (kinematics in our case) of the vehicle described by a system of differentially "flat" equations [22]. Consequently, all optimization occurs in the output space as opposed to the control space. This feature, when examined in a virtual domain, allows for fast prototyping of optimal trajectories. Additionally, since the IDVD method uses so few varied parameters, the computational requirements that must be met are significantly diminished. The method is also easy to modify and code, which gives the operator more freedom with regard mission scenarios.

B. IDVD METHOD COMPONENTS

1. Reference Trajectory

An important feature of the IDVD method is the fact that the reference trajectory is independent of any time constraints. This is accomplished by creating a path from mathematical functions for each Cartesian coordinate using a virtual arc "τ" as the independent variable. The equations for the x-coordinate are shown below.

$$x_i(\tau) = \sum_{k=0}^{n} a_{ik} \frac{(\max(1, k-2))! \tau^k}{k!}$$

$$\dot{x}_i(\tau) = \sum_{k=1}^{n} a_{ik} \frac{(\max(1, k-2))! \tau^{k-1}}{(k-1)!}$$

$$\ddot{x}_i(\tau) = \sum_{k=2}^{n} a_{ik} \tau^{k-2} \tag{35}$$

$$\dddot{x}_i(\tau) = \sum_{k=3}^{n} (k-2) a_{ik} \tau^{k-3}$$

The y coordinate is found through the same procedure. As seen, the simplicity of the Qbot allows for the use of simple polynomials of order "n". There are many different options for the choice of reference function and the type depends primarily on the shape of the trajectory that is desired. Curvy trajectories might be better approximated by a combination of lower-order polynomials and trigonometric terms. The degree of these polynomials is based on the number of boundary conditions specified by the problem and the coefficients a_{ik} must be determined algebraically. For example, the higher the time derivative of a vehicle at its beginning and final coordinates, the higher the order of the polynomial. The minimum polynomial degree is given by

$$n = d_o + d_f + 1 \tag{36}$$

d_o and d_f are the maximum orders of the time derivatives of the vehicle at the initial and end points. To clarify, if one were to specify the first and second time derivatives of the initial and final points of the vehicle, the order of the reference polynomial would be n=2+2+1=5.

However, to ensure there is flexibility with regard to the shape of the trajectory as well as smooth transitioning at the initial and final points, the third order

derivative, or jerk, are specified as an additional varied parameter. This amounts to three initial and three final conditions which according to Equation (36) would yield a seventh order polynomial. To solve for the coefficients of these polynomials, the following matrix equation can be set up.

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 & \frac{1}{30}\tau_f^6 & \frac{1}{42}\tau_f^7 \\
0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{3}\tau_f^3 & \frac{1}{4}\tau_f^4 & \frac{1}{5}\tau_f^5 & \frac{1}{6}\tau_f^6 \\
0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 & \tau_f^4 & \tau_f^5 \\
0 & 0 & 0 & 1 & 2\tau_f & 3\tau_f^2 & 4\tau_f^3 & 5\tau_f^4
\end{bmatrix}
*
\begin{bmatrix}
a_{x0} \\ a_{x1} \\ a_{x2} \\ a_{x3} \\ a_{x4} \\ a_{x5} \\ a_{x6} \\ a_{x7}
\end{bmatrix}
=
\begin{bmatrix}
x_0 \\ \dot{x}_0 \\ \ddot{x}_0 \\ \dddot{x}_0 \\ x_f \\ \dot{x}_f \\ \ddot{x}_f \\ \dddot{x}_f
\end{bmatrix}
\tag{37}
$$

The matrix equation for the y coordinates looks similar to the one below and has been omitted. In these equations x_0, $\dot{x}_0$, $\ddot{x}_0$, x_f, $\dot{x}_f$, $\ddot{x}_f$ are given while $\dddot{x}_0$ and $\dddot{x}_f$ are considered to be varied parameters.

2.   Speed Factor

As mentioned above, one of the advantages of the IDVD method is the ability to decouple space and time. This is why a virtual arc is used as an argument as opposed to time. This virtual arc calls for a virtual speed.

$$
\lambda = \frac{d\tau}{dt} \tag{38}
$$

In order to maintain simplicity, the virtual speed or speed factor is approximated with the same method that was used for the reference trajectory.

$$\lambda = \sum_{k=0}^{n} a_{\lambda k} \tau^{k} \frac{(\max(1, k-2))!}{k!}$$

$$\frac{d\lambda}{d\tau} = \sum_{k=1}^{n} a_{\lambda k} \tau^{k-1} \frac{(\max(1, k-2))!}{(k-1)!} \qquad (39)$$

$$\frac{d^2\lambda}{d\tau^2} = \sum_{k=2}^{n} a_{\lambda k} \tau^{k-2}$$

If we set the boundary conditions to unity such that $\lambda_0 = \lambda_f = 1$, the first derivatives to zero, and use the second derivatives as varied parameters, the matrix form can be determined as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 \\ 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{3}\tau_f^3 & \frac{1}{4}\tau_f^4 \\ 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 \end{bmatrix} \begin{bmatrix} a_{\lambda 0} \\ a_{\lambda 1} \\ a_{\lambda 2} \\ a_{\lambda 3} \\ a_{\lambda 4} \\ a_{\lambda 5} \end{bmatrix} = \begin{bmatrix} \lambda_0 \\ \lambda_0' \\ \lambda_0'' \\ \lambda_f \\ \lambda_f' \\ \lambda_f'' \end{bmatrix} \qquad (40)$$

3. Mapping from the Virtual to the Time Domain

Now that our candidate trajectory is defined in the virtual domain by Equation (35) (with coefficients found from Equation [37]), we need to produce time stamps or, in other words, transfer this trajectory to the physical (time) domain. Virtual argument τ varies from 0 to τ_f (another varied parameter), and computations are performed

at two hundred points along this virtual arc with the constant $\Delta\tau$. In between two nodes, τ_j and τ_{j+1}, the vehicle travels:

$$\Delta s = \sqrt{(x_{j+1}-x_j)^2+(y_{j+1}-y_j)^2} \tag{41}$$

For this interval, we also know λ_j and λ_{j+1}. Therefore, we also know that:

$$\Delta t = t_{j+1}-t_j = \frac{\Delta\tau}{\dfrac{\lambda_{j+1}+\lambda_j}{2}} \tag{42}$$

4. Cost Function

The cost function is constructed for usage with the unconstrained optimization Matlab routine, fminsearch. This function consists of a performance index with a minimum time or fixed time and also several penalty terms. Two terms penalize for violations of the Vmax constraint:

$$\begin{aligned} \max\left(0,\frac{\max_j(V_{left_j})-V_{max}}{V_{max}}\right)^2 \\ \max\left(0,\frac{\max_j(V_{right_j})-V_{max}}{V_{max}}\right)^2 \end{aligned} \tag{43}$$

Also, there is penalty turn for running into obstacles:

$$\max\left(0,\frac{D-\min_k\min_j d_{kj}}{D}\right)^2 \tag{44}$$

In this last term, d_{kj} is the distance at node j to obstacle k (index k runs through all K obstacles and j runs through all nodes), and D is the minimum distance.

Once a candidate trajectory is computed there is a cost attached to it. The fminsearch function then attempts to minimize this cost function by varying seven problem parameters $\ddot{x}_0, \ddot{x}_f, \ddot{y}_0, \ddot{y}_f, \ddot{\tau}_0, \ddot{\tau}_f, \tau_f$. To be more precise, it varies $\tau_0, \tau_f, \rho_0, \rho_f, \ddot{\tau}_0, \ddot{\tau}_f, \tau_f$ where

$$\begin{aligned}
\ddot{x}_0 &= \tau_0 \cos \rho_0 \\
\ddot{y}_0 &= \tau_0 \sin \rho_0 \\
\ddot{x}_f &= \tau_f \cos \rho_f \\
\ddot{y}_f &= \tau_f \sin \rho_f
\end{aligned}$$

(45)

THIS PAGE INTENTIONALLY LEFT BLANK

VII. LAB IMPLEMENTATION

A. INTRODUCTION

One of the main goals for this research is to provide the framework for a method of rapidly calculating a vehicle trajectory based on global information about the environment surrounding the vehicle. In this thesis, the global information is provided by a visual camera mounted on an aerial vehicle. The trajectory generation algorithm that is described above would certainly support information provided by other methods and sensors.

As a validation of the trajectory generation results, a test case involving the same vehicle and the same obstacle course is performed using reactive navigation. This method relies on sensor feedback from five infrared sensors mounted on the front half of the Qbot to create a vector potential field as discussed previously.

B. SIMULATION RESULTS

Due to the short duration of the flight and the fact that the obstacles are stationary in the test situations, the optimal trajectory is only calculated once on the ground station computer and then uploaded to the Qbot via a wireless ad-hoc network. The developed algorithm could most certainly be applied to a test where the optimal trajectory is recalculated during vehicle operation. This would help account for controller errors, disturbances, and dynamic obstacles.

First, the performance of the Qball as a platform for the downward-looking camera must be evaluated. At the start

of the trajectory generation test, the Qball-X4 is sent to
the position directly above the Qbot to acquire a photo.
The plots of the commanded Qball trajectory parameters
versus the actual parameters are given below:
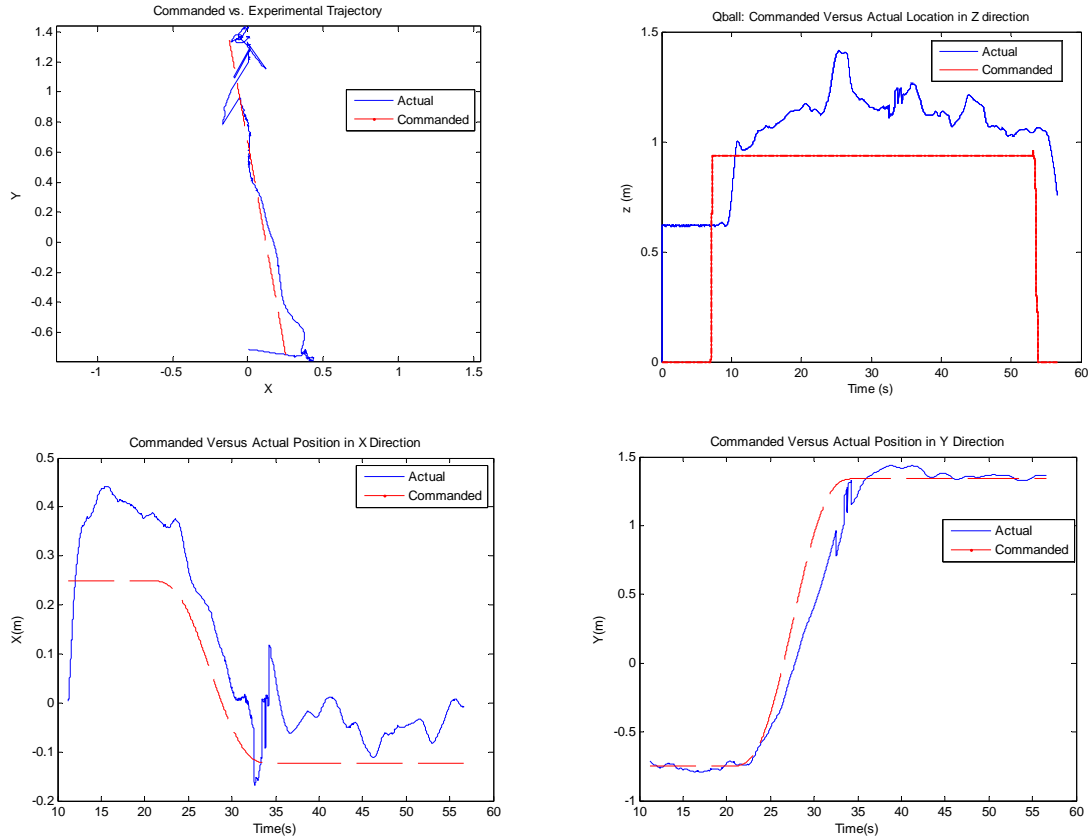
One of the issues that was discovered during the course of
this thesis was that the various controllers developed by
Quanser for the vehicle X-Y position and heights were not
optimized. As is evident from the graphs, there is a
significant amount of error in the tracking of the
commanded trajectory. Since the X-Y position of the
quadrotor is changed via modifications in roll and pitch, a
steady downward angle is difficult to achieve. In order to
take accurate pictures with reproducible image analysis
results, a stable platform is required. Additionally, the

Qball-X4 has only one processor on-board the vehicle. Simultaneous image processing and vehicle control proved to be too computationally intensive for the on-board computer. As a result, when calculating the optimal collision-free trajectory, the six obstacle positions were taken from measurements from the OptiTrack system.

As a benchmark to compare our optimal trajectory results against, the scenario was first run using only reactive navigation. Table 1 is a list of the initial coordinates for both the robot and the obstacle course (Figure 20).

	X Position(m)	Y Position(m)
Initial Point	0.1237	1.3417
Final Point	-0.024	-0.782
Obstacle 1	-0.5084	-0.2830
Obstacle 2	-0.3457	0.2553
Obstacle 3	-0.5174	0.7289
Obstacle 4	0.3056	-0.4343
Obstacle 5	0.4221	0.1804
Obstacle 6	0.3629	0.7832

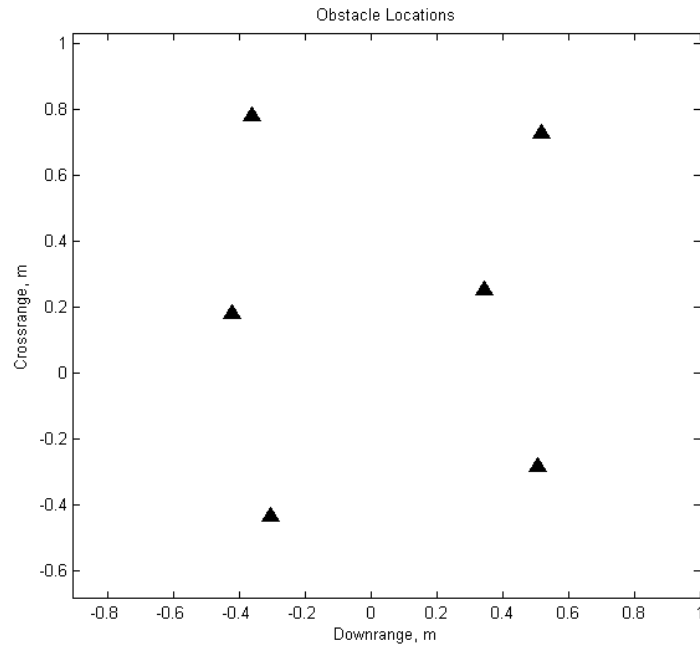Table 3.   Initial Robot and Obstacle Locations

Figure 20:   Obstacle Locations

The time to complete the course using reactive navigation was 70.3 seconds. Multiple tests were performed and the Qbot could not execute the course without collisions with max wheel speed greater than 0.05 m/s. Figure 20 shows a plot of the Qbot trajectory as well as the IR sensor returns:
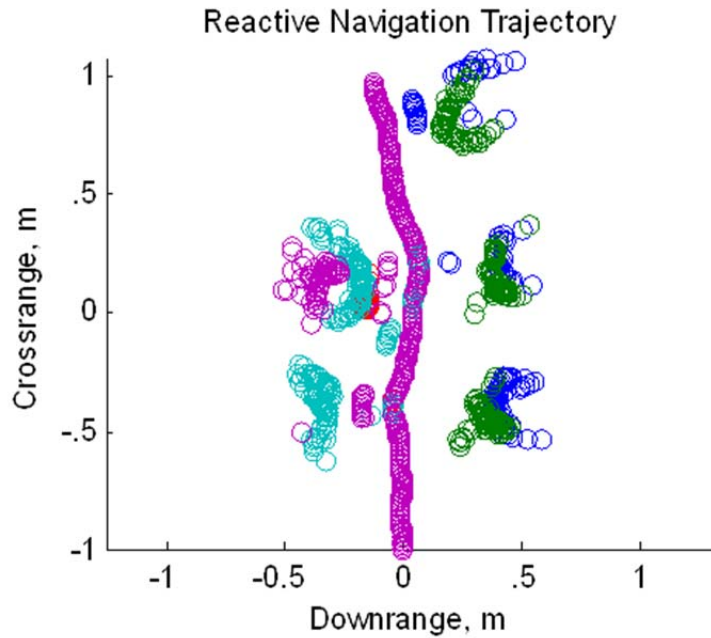
Figure 21:   Reactive Navigation Plot

As is evident from the plot, the trajectory, while collision-free, is certainly not optimal, and takes too much time to complete.

As a comparison, Figure 25 presents the result of the collaborative scenario where the Qbot utilizes the global information provided by the Qball. The trajectory only takes 26.4232 CPU seconds to compute and 15 seconds to complete. Figures 23-25 show the time histories of λ, heading, yaw rate, speed, and controls. In these plots, the last points for the heading and yaw rate are irrelevant because the speed is zero.
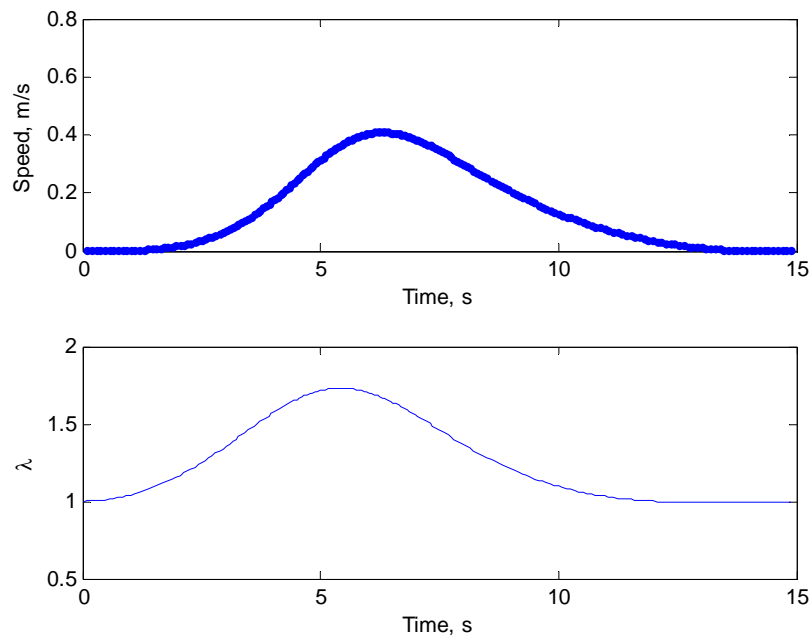
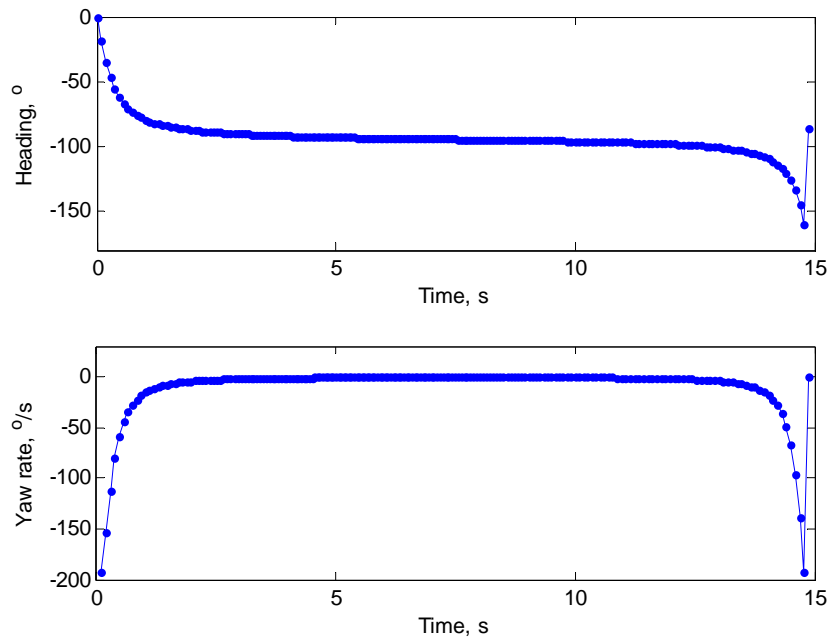Figure 22:   Physical and Virtual Speed Versus Time
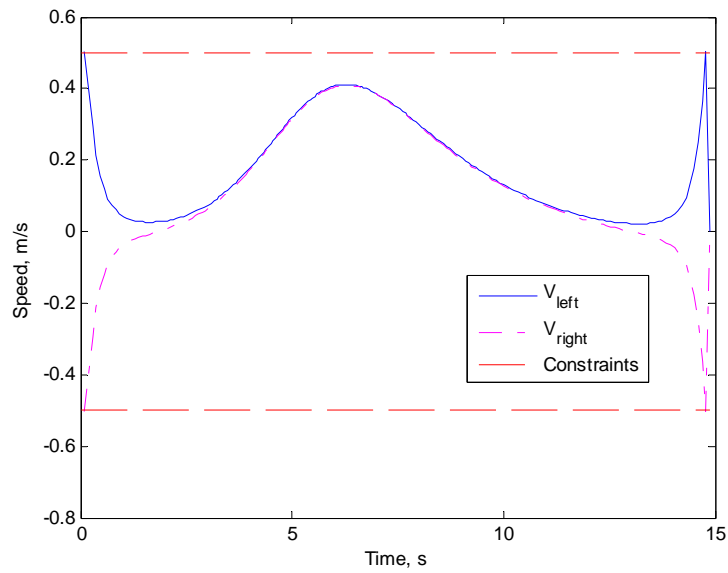

Figure 23:   Heading and Yaw Rate Versus Time

64

Figure 24:   Individual Wheel Speeds Versus Time

This solution is then fed to the Qbot controller (we use the interpl() function to produce a vector of control inputs that are evenly spaced in the time domain) and the Qbot tracks it perfectly.
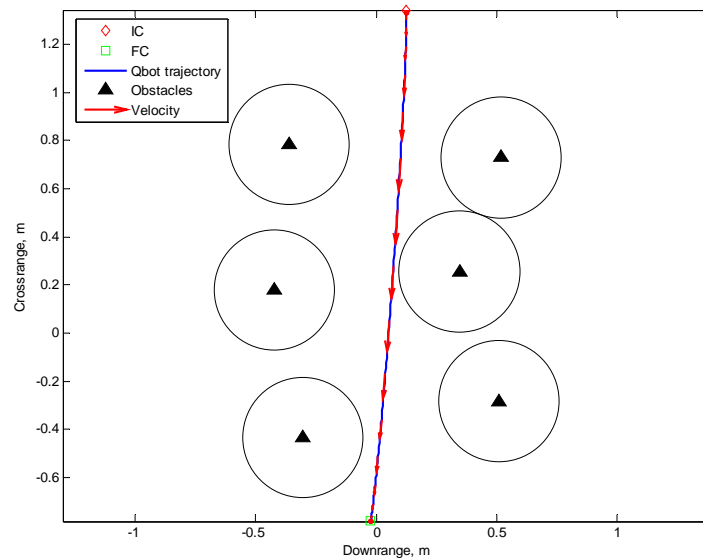

Figure 25:   IDVD Scenario #1 Trajectory

65

The trajectory shown in Figure 25 seems obvious but it should be emphasized that it also contains feasible controls and therefore is ready to be tracked.

A second scenario that may be deemed more sophisticated is shown in Figure 26.
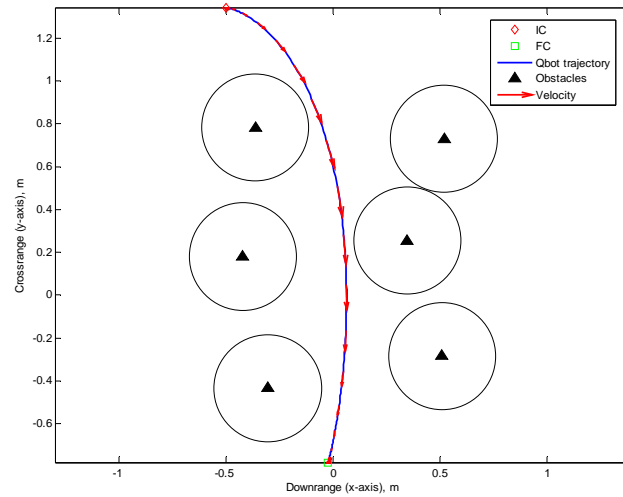

Figure 26:   IDVD Scenario #2 Trajectory
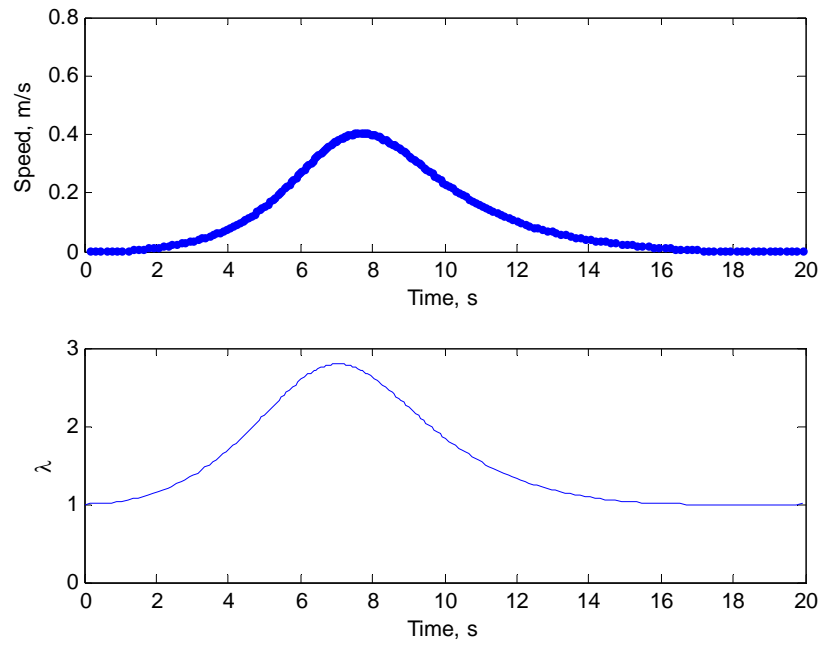
Figure 27:   Scenario #2 Virtual and Physical Speed
               Versus Time
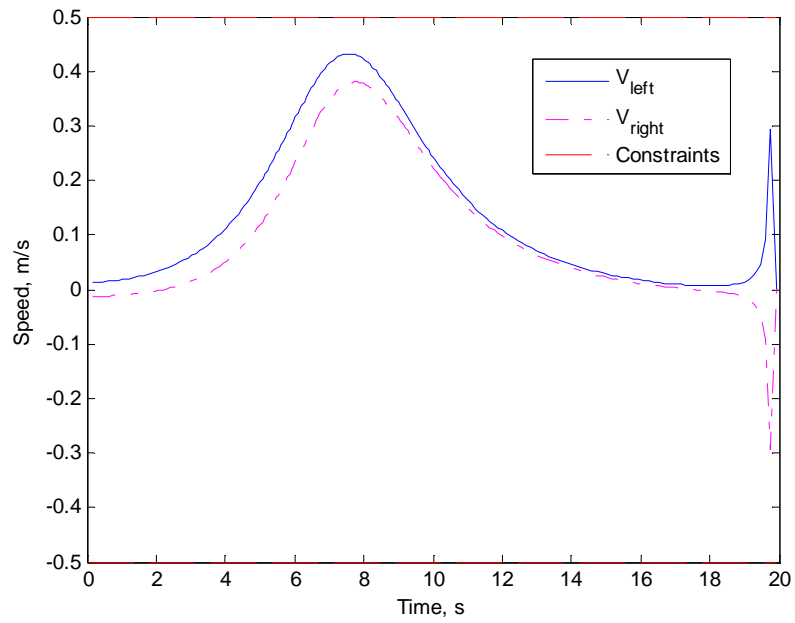
Figure 28:   Scenario #2 Individual Wheel Speeds Versus
               Time
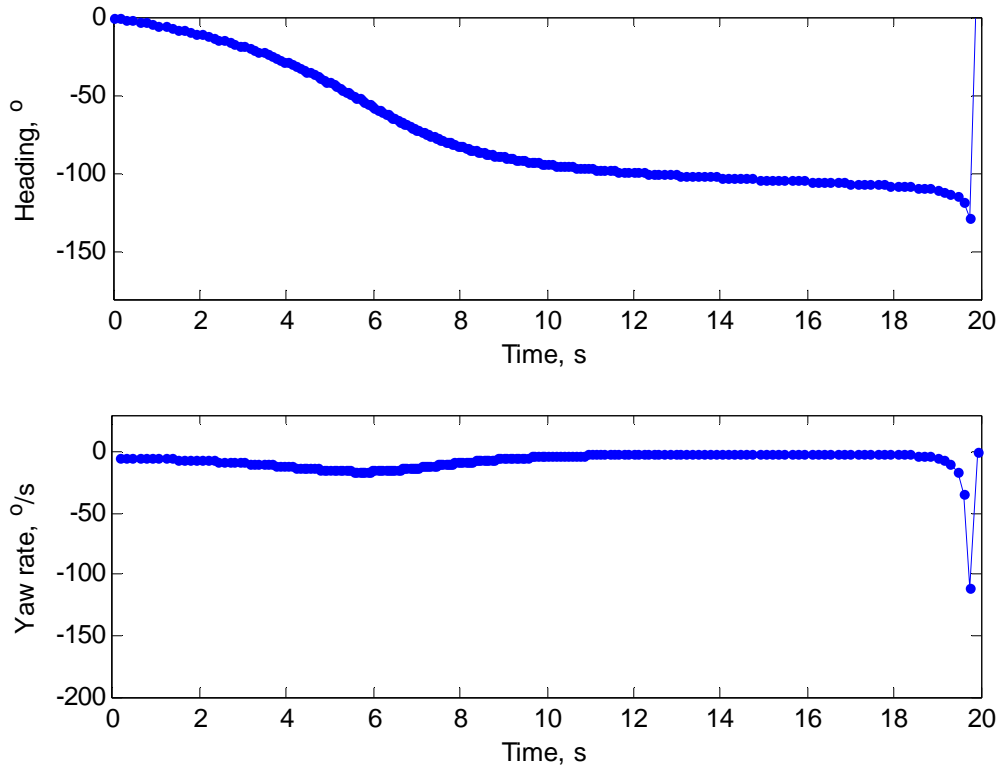
Figure 29:   Scenario #2 Heading and Yaw Rate Versus Time

Figures 27-29 show λ, heading, yaw rate, speed, and controls versus time. The trajectory here is curvier but the rest is the same—all the constraints are satisfied. In this case, the native reactive navigation Qbot controller had even more difficulty finding a way around. This example highlights the advantage of the proposed approach.

VIII. CONCLUSION AND FUTURE WORK RECOMMENDATION

A.   CONCLUSIONS

The following conclusions have been made through the course of this thesis research:

- An architecture for the control of multiple, varied unmanned vehicles has been developed and tested. This system allows for control via a single base station with minimal user input.

- The inverse dynamics in the virtual domain based trajectory generator is able to generate an accurate, feasible trajectory that allows for collision-free operation. This trajectory is faster and more accurate than reactive navigation methods.

- The framework for a system that recalculates the trajectory at multiple points along a path has been developed. This would allow for safe trajectories despite disturbances, controller inconsistencies, and dynamic obstacles.

- The native Qball LQR controllers do not accurately track trajectories. Tuned controllers would enable more options with regard to leader-follower missions involving both the quadrotor and ground vehicle.

- The Logitech camera on board both the Qball and Qbot has a slow rate of response. This makes image processing very challenging and requires very slow vehicle velocities in order to be useful.

B.   RECOMMENDATIONS

The following are recommendations for future work involving the Qball-X4 and Qbot unmanned systems:

- Improve the native Quanser controllers onboard the Qball-X4 quadrotor. This would allow the Qball to track trajectories with more precision and allow for a more stable platform for image capture.

- Utilize a second processor onboard the Qball-X4 for image capture and processing. This would free up the current embedded processor to control the vehicle.

- Use an improved reference function that would allow for more freedom with regard to trajectories. This could include a function that uses lower-order polynomials or trigonometric functions.

- Enable trajectory updates onboard the Qbot through the use of compiled C++ code.

- Incorporate more vehicles that must navigate through more complicated obstacle fields. This could include adding multiple quadrotors that would take pictures at various locations as well as multiple ground vehicles that must rendezvous at a certain point.

LIST OF REFERENCES

[1] E. Pastor, "UAV payload and mission control hardware/software architecture," *Aerospace and Electronic Systems Magazine,* vol. 22, no. 6, pp. 3–8, June 2007.

[2] G. Cai et al., "Introduction," in *Unmanned Rotorcraft Systems.* London, England: Springer, 2011, pp. 1–4.

[3] L. Feng et al., "Systematic design methodology and construction of UAV helicopters," *Mechatronics,* vol. 18, no. 10, pp. 547–48, December 2008.

[4] L. Feng et al., "Development of a vision-based ground target detection and tracking system for a small unmanned helicopter," in *Science in China Series F*, London, England: Springer, 2009.

[5] M. Turpin et al., "Trajectory design and control for aggressive formation flight with quadrotors," in *Autonomous Robots,* pp. 1–14, February 2012.

[6] D. Mellinger et al., "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *The International Journal of Robotics Research,* vol. 31, no. 5, pp. 664–674, April 2012.

[7] W. Etter et al., "Cooperative flight guidance of autonomous unmanned aerial vehicles," in *CPS Week Workshop on Networks of Cooperating Objects,* Chicago, IL, 2011, pp. 1–2.

[8] N. Vandapel et al., "Unmanned ground vehicle navigation using aerial ladar data," in *International Journal of Robotics Research*, vol. 25, no. 1, pp. 31–51, January 2006.

[9] V. Kumar et al., "Adaptive teams of autonomous aerial and ground robots for situational awareness," in *Journal of Field Robotics*, vol. 24, no. 11, pp.991–1014, November 2007.

[10] Quanser, "Quanser Unmanned Vehicle System Product Information Sheet," http://www.quanser.com/english/downloads/products/UVS_Labs/Quanser_UVS_PIS.pdf, accessed June, 2012.

[11] Quanser, "12 Things You Should Know About UVS," http://quanser.blogspot.com/2010/11/12-things-you-should-know-about-uvs.html, accessed May 2012.

[12] Maxbotix, "XL-MaxSonar-EZ3 Data Sheet," http://www.maxbotix.com/documents/MB1230-MB1330_Datasheet.pdf, accessed June, 2012.

[13] *Qbot User Manual*, Document Number 831, Quanser Academic, Toronto, Canada, pp. 4-9.

[14] NaturalPoint, "Optitrack," www.optitrack.com, accessed June, 2012.

[15] ETH Pixhawk, "MAV Computer Vision," https://pixhawk.ethz.ch/micro_air_vehicle/quadrotor/cheetah, accessed June, 2012.

[16] Quanser Innovate Educate, "Quanser Qball-x4 user manual," Document Number 888.

[17] X. Jing, *Mobile Robots Motion Planning: New Challenges*. Vienna, Austria: I-Tech, 2008. pp. 469-70.

[18] Quanser, "Qbot Motion Planning," Document Number 833, pp. 1-9.

[19] T. Zhang and Y Zhu, "Real-time motion planning for mobile robots by means of artificial potential field method in unknown environment," in *Industrial Robot: An International Journal*, vol. 37, no. 4, pp. 384-400.

[20] J. Park and J. Han, "Euclidian reconstruction from contour matches," in *Pattern Recognition*, vol. 25, no. 10, pp. 2109-2124.

[21] C. Hargraves and S. Paris, "Direct trajectory optimization using nonlinear programming and collocation", *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4., 1987, pp. 338-342

[22] O. Yakimenko et al., "Direct method based control system for an autonomous quadrotor," in *Journal of Intelligent & Robotic Systems*, vol. 60, no. 2,2010, pp. 285–316.

[23] G.Milionis, "A framework for collborative quadrotor-ground robot missions," M.S. thesis, Mech. Engr., NPS, Monterey, CA, 2011.

THIS PAGE INTENTIONALLY LEFT BLANK

Matlab Code

DMlopt.m-This code uses the function fminsearch and DMlfun to simulate the Simulink model DM2 in order to create an optimized trajectory that minimizes the specified costs.

```
close all, clear all, clc
warning off
%% Obstacles
% Obs=[1 2 0; -2 3 0; 2 3 0; 4 4 0];
% Obs=[441.6 189.7 312.4 -512.0 -593.8 -405.1; 0.25 0.25 0.25 0.25 0.25
0.25; -382.5 82.6 544.1 463.0 -206.1 -543.1]'/100;
% Obs=Obs(:,[1 3 2]);
Obs=[-0.5084 -0.3457 -0.5174 0.3056 0.4221 0.3629; -0.2830 0.2553
0.7289 -0.4343 0.18039 0.7832; 0.19 0.19 0.19 0.19 0.19 0.19]';
Obs(:,1)=-Obs(:,1);
%Obs=Obs(:,[1 3 2]);
%% Initial guesses for varied parameter
x0=[0.084       % Lambda dprime 0
   -0.0071      % Lambda dprime f
    0.0014      % X0ppp
    0.0012      % X0pppAngle
   -0.0012      % Xfppp
    0.0001      % XfpppX0pppAngle
   18/1000];    % tau f
%%
t = cputime;
options=optimset('TolFun',1e-3,'TolX',1e-
3,'Display','iter','MaxFunEvals',200);
[xv,fval,exitflag]=fminsearch(@DMlfun,x0,options);
time_elapsed = cputime-t
disp([x0 xv])
%%
x0=xv;
lam0_2pr     =x0(1);
lamf_2pr     =x0(2);
X0_tpl_prime =x0(3);
X0_tpl_primeA =x0(4);
Xf_tpl_prime =x0(5);
Xf_tpl_primeA =x0(6);
tauf         =x0(7);
```

DMlfun.m-This function provides the varied parameters and simulates the DM2 Simulink Model.

```
function f = DMlfun(x)
    lam0_2pr     =x(1);
    lamf_2pr     =x(2);
    X0_tpl_prime  =x(3);
    X0_tpl_primeA =x(4);
    Xf_tpl_prime  =x(5);
    Xf_tpl_primeA =x(6);
    tauf         =x(7);
opt=simset('SrcWorkspace','Current');
sim('DM2',[0 200], opt);
f=yout(length(yout));
```

Compute_Controls_pos.m- This code computes the commands for the Qbot ground vehicle and interpolates them so that they are provided to the controller at the right frequency.

```
% Controller speed
ctrl_t_step = 0.1;

% Run Simulation to get data
%sim('DM2', [0 200])
sim('DM3', [0 200])
[m,n] = size(a);
t_end = a(m,1);
t = 0:ctrl_t_step:t_end;

% Setup Variables
tau = a(:,1);
phi = a(:,2);
% theta = a(:,3);
x = a(:,3);
y = a(:,4);
z = a(:,5);
x_vel = a(:,6);
y_vel = a(:,7);
z_vel = a(:,8);
x_accel = a(:,9);
y_accel = a(:,10);
z_accel = a(:,11);
Vleft = a(:,12);
Vright = a(:,13);
lambda = a(:,14);


Vleft_mm=interp1(tau,Vleft,linspace(0,15,1+t_end/0.1),'spline')*1000;
```

```matlab
Vright_mm=interp1(tau,Vright,linspace(0,15,1+t_end/0.1),'spline')*1000;

Vleft_mm=[t; Vleft_mm]';
Vright_mm=[t; Vright_mm]';

% Vleft_mm=int16(Vleft_interp*1000);
% Vright_mm=int16(Vright_interp*1000);


%%
figure
plot(a(:,1),Vleft);
hold on;
title('Wheel Velocity Commands')
plot(a(:,1),Vright,'r');
plot([a(1,1) a(end,1)],.5*[1 1],'--r')
legend('V_{left}','V_{right}','Constraints',0)
plot([a(1,1) a(end,1)],-.5*[1 1],'--r')
xlabel('Time, s'), ylabel('Speed, mm/s')


%% Setup data for use in controller
% Setup a series of commands for the first waypoint
t_start = 0; %Start time for maneuver
t = t+t_start;

t_beginning = 0:ctrl_t_step:t_start-ctrl_t_step;
z_comp = ones(1,length(t_beginning));

% t_comp = [t_beginning' t_beginning';t' t'];
% x_command = [t_beginning' x(1)*z_comp';t' x'];
% y_command = [t_beginning' y(1)*z_comp';t' y'];
% z_command = [t_beginning' z(1)*z_comp';t' z'];

%% plot vehicle trajectory, superimpose obstacles
figure
title('Position of Qbot')
plot(x(1),y(1),'dm'), hold, plot(x(end),y(end),'rs'), plot(x,y,'b-')
xlabel('Downrange, m'), ylabel('Crossrange, m')
%daspect([1 1 1])  % set aspect ratio so circles appear as circles
hold on; axis equal
plot(Obs(:,1),Obs(:,2),'LineStyle','none','Marker','^',...
    'MarkerFaceColor','k','MarkerEdgeColor','k','MarkerSize',10)
legend('IC','FC','Qbot Trajectory','Obstacles',0)
% Plot the safety distance of 0.25m around each of the obstacles.
for i=1:length(Obs)
    rectangle('Position',[Obs(i,1)-0.25 Obs(i,2)-0.25 0.5 0.5],...
        'Curvature',[1 1],'EdgeColor','k')
end

%% Interpolate data
% Interpolate data between points at the same frequency the controller
% runs at.
%phi = interp1(tau,phi,t,'pchip');
```

77

```matlab
%theta = interp1(tau,theta,t,'pchip');
x = interp1(tau,x,t,'pchip');
y = interp1(tau,y,t,'pchip');
z = interp1(tau,z,t,'pchip');
x_vel = interp1(tau,x_vel,t,'pchip');
y_vel = interp1(tau,y_vel,t,'pchip');
z_vel = interp1(tau,z_vel,t,'pchip');
x_accel = interp1(tau,x_accel,t,'pchip');
y_accel = interp1(tau,y_accel,t,'pchip');
z_accel = interp1(tau,z_accel,t,'pchip');
figure
plot(tau,lambda)
xlabel('\tau')
ylabel('\lambda')
%
```

Controller_design.m- This code takes into account the Qball vehicle parameters in order to calculate the controller parameters.

```matlab
% This file contains all the controller parameters and LQR gains for the
% Qball.

% PITCH and ROLL
wnom = 15;
L = 0.2;
w = wnom;
K = 120;
J = 0.03;
Jyaw = 0.04;
CLimit = 0.025;
M = 1.4;
g = 9.8;

Am = [0 1 0
      0 0 2*K*L/J
      0 0 -w];
Bm = [0 0 w]';
Aobs = Am' ;
Bobs = eye(3);
Qobs = diag([.001 10000 .01]);

Robs = diag([ 1 1 1 ])*1;
Kobs = lqr(Aobs,Bobs,Qobs,Robs)
Kobs = Kobs';
Aobs = Aobs'-Kobs*Bobs';
eig(Aobs)
Bobs = [Bm Kobs]
Cobs = eye(3)
```

```matlab
Dobs = [ 0 0 0 0
    0 0 0 0
    0 0 0 0];

% augment with integrator
Ai = [Am [0 0 0 ]'
    1 0 0 0 ];
Bi = [Bm' 0]';
Ci = eye(4);
Di = [0 0 0 0 ]';
Q = diag([100 0 22000 10]);
R = 30000;

ki = lqr(Ai,Bi,Q,R);
rp_eig = eig(Ai-Bi*ki);
fprintf ('********************************************** \n');
fprintf('ROLL, PITCH DESIGN \n');
fprintf( 'P = %5.3f D = %5.3f Actuator = %5.3f I = %5.3f \n\n',ki(1),
ki(2),ki(3),ki(4));
for i = 1:4
fprintf('  %5.3f + %5.3f i \n  ',real(rp_eig(i)), imag(rp_eig(i)));
end;


%POSITION CONTROLLER (C2)
% XZ travel

tlimit = 5*pi/180; %max pitch cmd radians
%tlimit = 15*pi/180; %max pitch cmd radians
vlimit = 0.3; % max speed cmd in m/sec
%vlimit = 0.5; % max speed cmd in m/sec
Tau_theta = 1/7; % closed loop time constant for pitch response
wt =1/Tau_theta; %closed loop theta bandwidth
kt = 1;
a = [0 1 0 0
    0 0 g 0
    0 0 -wt 0
    1 0 0 0 ];
b = [0 0 wt 0 ]';

q = diag([ 5 2 0 0.1]);
%q = diag([ 5 2 0 0.1]);
%r = 50;
r = 50;

k = lqr(a,b,q,r);

ac = a-b*k;
xy_eig = eig(a-b*k);
Kp = k(1);
Kd = k(2);
Ki = k(4);
Kw = k(3);
fprintf('\n\n X Y Design \n');
```

```matlab
fprintf( 'P = %5.3f D = %5.3f Actuator = %5.3f I = %5.3f \n\n',k(1),
k(2),k(3),k(4));
for i = 1:4
fprintf('  %5.3f + %5.3f i \n  ',real(xy_eig(i)), imag(xy_eig(i)));
end;

% Z axis w actuator

% vlimith = 0.1;
% Amh = [0 1 0
%      0 0 4*K/M
%      0 0 -w];
% Bmh = [0 0 w]';
% Cmh = eye(3);
% Dmh = [0 0 0 ]';
%
% % augment with integrator
% Ahi = [Amh [0 0 0 ]'
%      1 0 0 0 ];
% Bhi = [Bmh' 0]';
% Chi = eye(4);
% Dhi = [0 0 0 0 ]';
% Q = diag([30 .8 12000 10]);
% Q = diag([0 0 000 100]);
% R = 2000000;
% kh = lqr(Ahi,Bhi,Q,R);
% rp_eig = eig(Ai-Bi*kh);
% fprintf ('********************************************** \n');
% fprintf('Z DESIGN \n');
% fprintf( 'P = %5.3f D = %5.3f Actuator = %5.3f I = %5.3f \n\n',kh(1),
kh(2),kh(3),kh(4));
% for i = 1:4
% fprintf('  %5.3f + %5.3f i \n  ',real(rp_eig(i)), imag(rp_eig(i)));
% end;
% Kph = kh(1);
% Kdh = kh(2);
% Kwh = kh(3);
% Kih = kh(4);


% Z axis without actuator

vlimith = 0.1;
Amh = [0 1
     0 0 ]
Bmh = [0 4*K/M]';
Cmh = [1 0];
Dmh = 0;

% augment with integrator
Aih = [Amh [0  0 ]'
     1  0 0 ];
Bih = [Bmh' 0]';
```

80

```
Cih = eye(3);
Dih = [0 0 0]';

Q = diag([1 0 50]);
R = 5000000;
kh = lqr(Aih,Bih,Q,R);
h_eig = eig(Aih-Bih*kh);
fprintf ('********************************************** \n');
fprintf('Z DESIGN \n');
fprintf( 'P = %5.3f D = %5.3f I = %5.3f \n\n',kh(1), kh(2),kh(3));
for i = 1:3
fprintf('  %5.3f + %5.3f i \n  ',real(h_eig(i)), imag(h_eig(i)));
end;
Kph = kh(1);
Kdh = kh(2);
Kwh = 0;
Kih = kh(3);

% yaw axis

Ky = 4;
Jy = 0.032;

Amy = [0 1
    0 0 ];
Bmy = [0 4*Ky/Jy]';
Cmy = eye(2);
Dmy = [0;0];
Qy = diag([1 0.1]);
Ry = 1000;
ky = lqr(Amy,Bmy,Qy,Ry);
h_eigy = eig(Amy-Bmy*ky);
Kpyaw = ky(1);
Kdyaw = ky(2);
Bih = [Bih,[0 1 0]'];
Dih = [Dih, [0 0 0]'];
```

Filter_design.m- This code is run when Qball-X4 model is first started in order to find the complimentary filter coefficients.

```
t=10;
s = tf('s');
Gg = t^2*s/(t*s+1)^2
Gi = (2*t*s+1)/(t*s+1)^2
```

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California